A Tractable Probabilistic Model for Subset Selection

Yujia Shen and **Arthur Choi** and **Adnan Darwiche**
Computer Science Department
University of California
Los Angeles, CA 90095
{yujias,aychoi,darwiche}@cs.ucla.edu

Abstract

Subset selection tasks, such as top-k ranking, induce datasets where examples have cardinalities that are known a priori. In this paper, we propose a tractable probabilistic model for subset selection and show how it can be learned from data. Our proposed model is interpretable and subsumes a previously introduced model based on logistic regression. We show how the parameters of our model can be estimated in closed form given complete data, and propose an algorithm for learning its structure in an interpretable space. We highlight the intuitive structures that we learn via case studies. We finally show how our proposed model can be viewed as an instance of the recently proposed Probabilistic Sentential Decision Diagram.

1 INTRODUCTION

We consider in this paper the problem of selecting k items from among a set of n alternatives. This *subset selection* problem appears in a number of domains, including resource allocation (what resources shall I allocate?), preference learning (which items do I prefer?), human computation (which labelers should I recruit for my task?), and sports (which players shall play the next game?). In these subset selection tasks, a dataset consists of examples in which a fixed number (k) of variables is set to true from among a total of n variables representing choices. Given such a dataset, the goal is to learn a generative probabilistic model that can accurately represent the underlying processes that govern these selections.

Commonly used representations such as Bayesian and Markov networks are not well-suited for learning from this type of data. In general, the underlying cardinality constraints would lead to fully-connected (and hence intractable) structures—hence more specialized representations are needed to model such subset selection tasks. Recently, a new type of probabilistic model was proposed, called the n-choose-k model (Swersky, Tarlow, Adams, Zemel, & Frey, 2012), that can take into account datasets whose examples have a known cardinality. The proposal includes a view on the popular logistic regression model as a mixture of n-choose-k models (with a component for each k from 0 to n). Both inference and learning are tractable in the n-choose-k model.

In this paper, we propose a more expressive model for subset selection processes, called the *recursive n-choose-k model,* which we derived from a more general tractable representation called the Probabilistic Sentential Decision Diagram (PSDD) (Kisa, Van den Broeck, Choi, & Darwiche, 2014). Our proposed model is *tractable* as it can accommodate a variety of probabilistic queries in polynomial time. It is also *highly interpretable* as its parameters have precise meanings and its underlying structure explicates a generative process for the data. This is in contrast to similar tractable representations such as Arithmetic Circuits (ACs) (Lowd & Domingos, 2008; Lowd & Rooshenas, 2013; Shen, Choi, & Darwiche, 2016; Bekker, Davis, Choi, Darwiche, & Van den Broeck, 2015) and their Sum-Product Networks (SPNs) variant (Poon & Domingos, 2011; Gens & Domingos, 2012; Dennis & Ventura, 2015). We propose a simplified closed-form parameter estimation algorithm for our recursive n-choose-k model, as well as a simple but effective structure learning algorithm. Empirically, we show how our recursive n-choose-k model is more expressive and can provide a better fit for datasets with known cardinalities than previously proposed models for this task.

This paper is organized as follows. In Section 2, we review the subset selection problem and a previously proposed model. In Section 3, we introduce our recursive n-choose-k model. In Section 4, we present the corresponding parameter and structure learning algorithms. In Section 5, we evaluate our model empirically, and present some case studies. In Section 6, we show how the proposed n-choose-k model corresponds to a PSDD for n-choose-k constraints. Section 7 closes with some concluding remarks. Proofs and additional experiments are provided in the supplementary appendix.

2 n-CHOOSE-k MODELS

As a running example, consider the subset selection problem that computer science (CS) students regularly face: selecting k out of n courses to take in a quarter. For simplicity, say that students select three out of the following six courses:

learning (ML)	computability (CP)	linear algebra (LA)
logic (LG)	complexity (CX)	calculus (CL)

By column, we have two AI classes, two CS theory classes, and two math classes.

Let us now consider a dataset over students and the courses that they select. We have six variables $(\texttt{ML}, \texttt{LG}, \texttt{CP}, \texttt{CX}, \texttt{LA}, \texttt{CL})$ representing the classes that a student can select, where $\texttt{ML}=1$ means the student selected machine learning, whereas $\texttt{ML}=0$ means they did not. Our dataset consists of examples such as:

$$\begin{aligned}&\texttt{ML}=0, \texttt{LG}=1, \texttt{CP}=1, \texttt{CX}=1, \texttt{LA}=0, \texttt{CL}=0\\ &\texttt{ML}=1, \texttt{LG}=1, \texttt{CP}=0, \texttt{CX}=0, \texttt{LA}=1, \texttt{CL}=0\\ &\texttt{ML}=1, \texttt{LG}=0, \texttt{CP}=0, \texttt{CX}=1, \texttt{LA}=0, \texttt{CL}=1.\end{aligned}$$

Since students must select three out of six classes, each example in the dataset has exactly three positive entries, and thus three negative entries as well. We refer to such a dataset as an *n-choose-k* dataset: each example has exactly k out of n variables appearing positively, where k is called the example *cardinality.* A CS department with student enrollment data of this type may want to analyze this data and reason about the courses that students choose to take.

In this paper, we assume that the cardinality k is known a priori. For example, in preference learning, we may be provided with data where users have selected their top-10 favorite movies, in which case exactly 10 variables appear positively, and the rest appear negatively.

(Swersky et al., 2012) proposed a probabilistic model, called the *n-choose-k* model, which assumes a prior over k. A simpler form is obtained when k is fixed, leading to the following distribution over a set of n variables $\mathbf{X}$:

$$Pr_k(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z_k(\boldsymbol{\theta})} \prod_{X \in \mathbf{X}} \exp\{\theta_X \cdot \lambda_X\} \tag{1}$$

for instantiations $\mathbf{x}$ with cardinality k; $Pr(\mathbf{x}) = 0$ otherwise. First, $\boldsymbol{\theta} = (\ldots, \theta_X, \ldots)$ is a vector of n parameters, one parameter θ_X for each variable $X \in \mathbf{X}$. Next, we have a vector $(\ldots, \lambda_X, \ldots)$ of n indicators, one indicator λ_X for each variable $X \in \mathbf{X}$, where λ_X is 1 if $\mathbf{x}$ sets X positively and 0 otherwise. Finally, $Z_k(\boldsymbol{\theta}) = \sum_{\mathbf{x}:\mathsf{Card}(\mathbf{x})=k} \prod_{X\in\mathbf{X}} \exp\{\theta_X \cdot \lambda_X\}$ is the normalizing constant, where $\mathsf{Card}(\mathbf{x})$ is the cardinality of $\mathbf{x}$. When we take a mixture of these models, for k from 0 to n, we obtain the class conditional distribution of the logistic regression model (Swersky et al., 2012):

$$Pr(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{X \in \mathbf{X}} \exp\{\theta_X \cdot \lambda_X\}$$

for all instantiations $\mathbf{x}$ (of any cardinality), where $Z(\boldsymbol{\theta}) = \prod_{X\in\mathbf{X}}(1 + e^{\theta_X})$. Hence, we refer to the model of Equation 1 as the *logistic n-choose-k* model.

Example 1. In our course selection problem, every course X has a parameter θ_X, where a larger parameter value corresponds to a higher course popularity. For example, the parameters $(\theta_{\texttt{ML}}, \theta_{\texttt{LG}}, \theta_{\texttt{CP}}, \theta_{\texttt{CX}}, \theta_{\texttt{LA}}, \theta_{\texttt{CL}}) = (3, 2, 1, -1, -2, -3)$ suggest that ML is the most popular course. The probability of a student selecting machine learning (ML), logic (LG) and computability (CP) is then

$$\begin{aligned}&Pr(\texttt{ML}=1, \texttt{LG}=1, \texttt{CP}=1, \texttt{CX}=0, \texttt{LA}=0, \texttt{CL}=0)\\ &= \tfrac{1}{Z}\exp\{\theta_{\texttt{ML}} + \theta_{\texttt{LG}} + \theta_{\texttt{CP}} + 0 + 0 + 0\} = \tfrac{1}{Z}\exp\{6\}\end{aligned}$$

where $Z \approx 529.06$ is a normalization constant. □

Next, we propose a more refined model that views course selection as a recursive process.

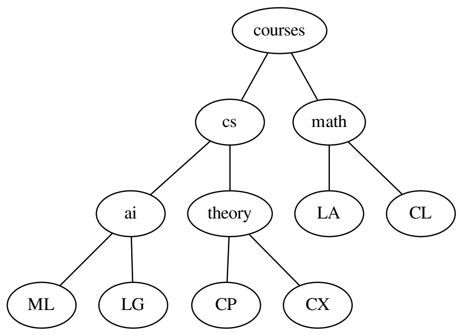

Figure 1: A tree hierarchy of courses.

3 RECURSIVE n-CHOOSE-k MODELS

Consider the following recursive process of course selection that a student may follow, which traces the tree structure of Figure 1. First, at the root courses, a student decides *how many* cs classes to take, compared to the number of math classes to take, out of a total of 3 classes. Say the student decides to take 2 cs classes and 1 math

Algorithm 1 $\mathsf{Sample}(v, k)$

```
input: Node $v$ in a vtree and a number $k$, $0 \leq k \leq |\mathbf{X}_v|$
output: A selection of $k$ variables from $\mathbf{X}_v$
main:
1: if $v$ is a leaf node labeled with variable $X$ then
2:     return $\{X=0\}$ if $k=0$ else return $\{X=1\}$
3: else
4:     $v_1, v_2 \leftarrow$ children of node $v$
5:     $\theta_{v,k} \leftarrow$ distribution over $(k_1, k_2)$ s.t. $k_1 + k_2 = k$
6:     $(k_1, k_2) \leftarrow$ a cardinality pair drawn from $\theta_{v,k}$
7:     return $\mathsf{Sample}(v_1, k_1) \cup \mathsf{Sample}(v_2, k_2)$
```

class. Following the left branch, the student decides *how many* classes to take, now between ai and theory. Suppose they take one ai class, and hence one theory class. The student then recursively decides between learning (ML) and logic (LG), and independently, between computability (CP) and complexity (CX). We backtrack and decide (independently of the prior choices) between linear algebra (LA) and calculus (CL).

Algorithm 1 describes a probabilistic generative process for subset selection, based on a tree structure similar to the one in Figure 1. This structure is called a variable tree, or *vtree,* and corresponds to a full, binary tree with each leaf node v labeled with a distinct variable $X \in \mathbf{X}$. For a vtree node v, we will use $\mathbf{X}_v$ to denote the set of variables appearing at or below node v. We will also use v_1 and v_2 to represent the left and right children, respectively, of an internal vtree node v.

A call to $\mathsf{Sample}(v, k)$ of Algorithm 1 randomly selects k variables from $\mathbf{X}_v$. If v is an internal node, we first sample a cardinality k_1 of variables to select from the left child of v (which implies that we select $k_2 = k - k_1$ variables from the right child of v). If v is a leaf node, then k is either 0 (we do not select a variable), or 1 (we select the variable X at node v).

This generative process describes our new probabilistic model of subset selection, which we call the *recursive n-choose-k model.* We formalize this model next.

3.1 FORMAL DEFINITION

From here on, we assume that $n \geq 1$ and $0 \leq k \leq n$.

To define our recursive n-choose-k model, we first need to define the notion of a *choice distribution*, for deciding how many elements to choose. Such a distribution is defined for three integers n_1, n_2 and k where $n_1, n_2 \geq 1$ and $0 \leq k \leq n_1 + n_2$. The domain of this distribution is the set of pairs (k_1, k_2) such that $k_1 \leq n_1$, $k_2 \leq n_2$ and $k_1 + k_2 = k$. The intuition behind a choice distribution is this. We need to select k items from $n_1 + n_2$ items. Each pair (k_1, k_2) corresponds to a choice of k_1 items from the n_1 items and a choice of k_2 items from the n_2 items. The n_1 and n_2 items will be the variables appearing in the left and right subtrees of a vtree node v; that is, $n_1 = |\mathbf{X}_{v_1}|$ and $n_2 = |\mathbf{X}_{v_2}|$. Hence, we will denote the parameters of a choice distribution by $\theta_{v,k}(k_1, k_2)$, which represents the probability that we will select k_1 items from the left subtree of v and k_2 items from the right subtree of v. This implies that $k \leq |\mathbf{X}_v|$.

Example 2. Consider the following choice distributions from our course selection problem (we are only showing choices with non-zero probability).

v	k	k_1, k_2	$\theta_{v,k}$
courses	3	$1, 2$	0.1
		$2, 1$	0.3
		$3, 0$	0.6
math	2	$1, 1$	1.0
math	1	$1, 0$	0.3
		$0, 1$	0.7
math	0	$0, 0$	1.0

v	k	k_1, k_2	$\theta_{v,k}$
cs	2	$0, 2$	0.3
		$1, 1$	0.6
		$2, 0$	0.1
cs	1	$1, 0$	0.4
		$0, 1$	0.6
ai	1	$1, 0$	0.4
		$0, 1$	0.6

For $v=$courses and $k=3$, the choice distribution $\theta_{v,k}$ is used to select 3 courses from cs and math. For example, we select 1 course from cs and 2 from math with probability $\theta_{v,k}(1, 2) = 0.1$. □

We are now ready to define our subset selection model. A recursive n-choose-k model over n binary variables $\mathbf{X}$ has two components: (1) *structure:* a vtree where each leaf node is labeled with a distinct variable from $\mathbf{X}$, and (2) *parameters:* for each internal vtree node v with m leaves, a choice distribution $\theta_{v,i}$ for each $i = \max(0, k - (n - m)), \ldots, \min(k, m)$. In a recursive n-choose-k model, we will never choose more than k items at any vtree node v, hence we need choice distributions for at most $\min(k, m)$ items at node v. Moreover, since we can choose at most $n - m$ items from outside node v, we must choose at least $k - (n - m)$ items at node v. Hence, we do not need choice distributions for fewer items than $\max(0, k - (n - m))$.

The distribution induced by a recursive n-choose-k model is defined inductively, over instantiations $\mathbf{x}$ whose cardinalities are k. Note that for the inductive cases, we refer to cardinalities by i rather than by k.

For the base case of a leaf vtree node v labeled with variable X, we have $Pr_{v,i}(X=\mathsf{true}) = 1$ if $i=1$ and 0 if $i=0$. For the inductive case of an internal leaf node v:

$$Pr_{v,i}(\mathbf{x}_v) = Pr_{v_1,i_1}(\mathbf{x}_{v_1}) \cdot Pr_{v_2,i_2}(\mathbf{x}_{v_2}) \cdot \theta_{v,i}(i_1, i_2).$$

Here, $\mathbf{x}_{v_1}$ and $\mathbf{x}_{v_2}$ are the subsets of instantiation $\mathbf{x}_v$ pertaining to variables $\mathbf{X}_{v_1}$ and $\mathbf{X}_{v_2}$, respectively. Moreover, i_1 and i_2 are the cardinalities of instantiations $\mathbf{x}_{v_1}$

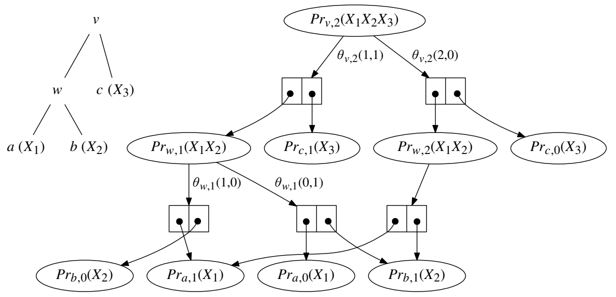

Figure 2: A vtree (upper-left), with a corresponding recursive 3-choose-2 distribution (right). Leaf vtree nodes are labeled with their variables inside parenthesis.

and $\mathbf{x}_{v_2}$, respectively (hence, $i_1 + i_2 = i$). The underlying independence assumption in the above inductive case is this: how we select i_1 elements from v_1 is independent of how we select i_2 elements from v_2, after we have chosen how many elements i_1 and i_2 to select in each.

Figure 2 depicts a vtree and a corresponding 3-choose-2 model. Each circled node represents a recursive n-choose-k model that is associated with an internal vtree node, for particular values of n and k. Each circled node is also associated with a choice distribution, whose parameters annotate the edges outgoing the node.

Example 3. Using the recursive n-choose-k model, and the choice distributions of Example 2, the probability that a student takes machine learning ($\texttt{ML}$), logic ($\texttt{LG}$) and linear algebra ($\texttt{LA}$) is

$$\begin{aligned} &Pr(\texttt{ML}{=}1, \texttt{LG}{=}1, \texttt{CP}{=}0, \texttt{CX}{=}0, \texttt{LA}{=}1, \texttt{CL}{=}0) \\ &= \theta_{\textsf{courses},3}(2,1) \cdot \theta_{\textsf{cs},2}(2,0) \cdot \theta_{\textsf{math},1}(1,0) \cdot \\ &\qquad \theta_{\textsf{ai},2}(1,1) \cdot \theta_{\textsf{theory},0}(0,0) \\ &= 0.3 \cdot 0.1 \cdot 0.3 \cdot 1 \cdot 1 = 0.009. \qquad \square \end{aligned}$$

Finally, we show that our recursive n-choose-k model subsumes the logistic n-choose-k model of Equation 1.

Proposition 1 *For any logistic n-choose-k model, there is a recursive n-choose-k model that induces the same distribution.*

3.2 TRACTABLE INFERENCE

Recursive n-choose-k models are *tractable* probabilistic models: we can perform many probabilistic queries in time linear in the size of the model. For example, we can compute the most probable explanation (MPE), the probability of evidence, and posterior marginals, all in linear time. For example, we can use MPE to extend a partial subset selection to a complete one (e.g., to extend a user's top-3 list of movies to a top-10 list of movies, to provide movie suggestions). We can perform cardinality queries efficiently: given a user's top-3 list, what is the expected number of comedies that would appear on their top-10 list? This tractability is inherited from the Probabilistic Sentential Decision Diagram (Kisa et al., 2014), of which the recursive n-choose-k model is a concrete example. We discuss this connection further in Section 6.

As an example, consider a recursive n-choose-k model and suppose we observed evidence $\mathbf{e}$ on some of its variables $\mathbf{E} \subseteq \mathbf{X}$. We can compute the probability of this evidence recursively, starting from the root vtree node v:

$$Pr_{v,i}(\mathbf{e}) = \sum_{\theta_{v,i}(i_1,i_2)} Pr_{v_1,i_1}(\mathbf{e}_{v_1}) Pr_{v_2,i_2}(\mathbf{e}_{v_2}) \theta_{v,i}(i_1,i_2),$$

which follows from the underlying independence assumptions. In the base case, v is a leaf vtree node with variable X, and $i \in \{0,1\}$. If the evidence $\mathbf{e}$ is empty, then $Pr_{v,i}(\mathbf{e}) = 1$. Otherwise, $Pr_{v_i}(\mathbf{e}) = 1$ iff evidence $\mathbf{e}$ and the 1-choose-i model sets X to the same value.

Example 4. Say we want to compute the probability that a student takes learning ($\texttt{ML}$) and linear algebra ($\texttt{LA}$) out of 3 total classes, with the choice distributions of Example 2. With evidence $\mathbf{e} = \{\texttt{ML} = 1, \texttt{LA} = 1\}$, we have:

$$\begin{aligned} &Pr_{\textsf{courses},3}(\texttt{ML}{=}1, \texttt{LA}{=}1) \\ &= Pr_{\textsf{cs},2}(\texttt{ML}{=}1) \cdot Pr_{\textsf{math},1}(\texttt{LA}{=}1) \cdot \theta_{\textsf{courses},3}(2,1) \\ &+ Pr_{\textsf{cs},1}(\texttt{ML}{=}1) \cdot Pr_{\textsf{math},2}(\texttt{LA}{=}1) \cdot \theta_{\textsf{courses},3}(1,2) \\ &= Pr_{\textsf{cs},2}(\texttt{ML}{=}1) \cdot 0.3 \cdot 0.3 + Pr_{\textsf{cs},1}(\texttt{ML}{=}1) \cdot 1 \cdot 0.1. \end{aligned}$$

Recursively, we compute $Pr_{\textsf{cs},2}(\texttt{ML}{=}1) = 0.34$ and $Pr_{\textsf{cs},1}(\texttt{ML}{=}1) = 0.16$, which yields:

$$Pr_{\textsf{courses},3}(\mathbf{e}) = 0.34 \cdot 0.09 + 0.16 \cdot 0.1 = 0.0466. \quad \square$$

4 LEARNING n-CHOOSE-k MODELS

We show in this section how to estimate the parameters of a recursive n-choose-k model in closed form. We also propose a simple structure learning algorithm for these models, which amounts to learning their underlying vtrees (i.e., the recursive partitioning of variables).

We first consider the number of parameters in a recursive n-choose-k model, which is relevant to our structure learning algorithm. Each leaf vtree node corresponds to a 1-choose-1 or a 1-choose-0 model, which has no parameters. There are $O(n)$ internal nodes in the vtree, and each one has $O(k)$ choice distributions associated with it. Each of these distributions has $O(k)$ parameters, leading to a total of $O(nk^2)$ parameters. Hence, the total number of parameters in a recursive n-choose-k model is bounded by a polynomial in n and k.

To contrast, there are n parameters in a logistic n-choose-k model, which can be learned by iterative methods such as gradient descent (Swersky et al., 2012). Moreover, unlike our recursive model, there is no structure to be learned in a logistic n-choose-k model.

4.1 PARAMETER LEARNING

Suppose we are given a set of n binary variables $\mathbf{X}$. Let $\mathcal{D}$ be a dataset containing N examples, where each example is an instantiation $\mathbf{x}$ of variables $\mathbf{X}$ with exactly k variables set to true (that is, $\mathcal{D}$ is an n-choose-k dataset).

For a set of variables $\mathbf{Y} \subseteq \mathbf{X}$, we will say that an example $\mathbf{x}$ has $\mathbf{Y}$-cardinality equal to m iff exactly m variables in $\mathbf{Y}$ are set to true in the example. We will also use $\mathcal{D}\#(\mathbf{Y}:m)$ to denote the number of examples in dataset $\mathcal{D}$ with $\mathbf{Y}$-cardinality equal to m. This allows us to define the following empirical probability, which is the probability of having $\mathbf{Y}$-cardinality equal to m:

$$Pr_{\mathcal{D}}(\mathbf{Y}:m) = \tfrac{1}{N}\mathcal{D}\#(\mathbf{Y}:m).$$

Example 5. Consider the example

$$\mathtt{ML}=0, \mathtt{LG}=1, \mathtt{CP}=1, \mathtt{CX}=0, \mathtt{CL}=0, \mathtt{LA}=1$$

which has an ai-cardinality of 1 and a cs-cardinality of 2. We can compute the empirical probability that a student takes one ai course and two cs courses by counting examples in the dataset:

$$Pr_{\mathcal{D}}(\mathsf{ai}:1, \mathsf{cs}:2) = \tfrac{1}{N}\mathcal{D}\#(\mathsf{ai}:1, \mathsf{cs}:2).$$

We can also find the conditional probability that a student takes one ai course given that they take two cs courses:

$$Pr_{\mathcal{D}}(\mathsf{ai}:1 \mid \mathsf{cs}:2) = \frac{\mathcal{D}\#(\mathsf{ai}:1, \mathsf{cs}:2)}{\mathcal{D}\#(\mathsf{cs}:2)}. \qquad \Box$$

The following theorem provides a closed form for the maximum likelihood estimates of a recursive n-choose-k model given a corresponding dataset $\mathcal{D}$.

Theorem 1 *Consider a recursive n-choose-k model and dataset $\mathcal{D}$, both over variables $\mathbf{X}$. Let v be an internal vtree node of this model. The maximum-likelihood parameter estimates for node v are unique and given by*

$$\begin{aligned}\theta_{v,i}(i_1, i_2) &= Pr_{\mathcal{D}}(\mathbf{X}_{v_1}:i_1 \mid \mathbf{X}_v:i) \\ &= Pr_{\mathcal{D}}(\mathbf{X}_{v_2}:i_2 \mid \mathbf{X}_v:i).\end{aligned}$$

According to this theorem, and given the underlying vtree of a recursive n-choose-k model, we can estimate its maximum likelihood parameters by performing a single pass on the given dataset.

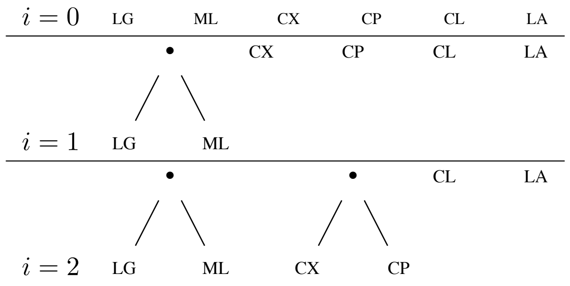

Figure 3: The first few iterations of vtree learning.

4.2 STRUCTURE LEARNING

We now turn to learning the structure of a recursive n-choose-k model, which amounts to learning its underlying vtree. Our approach will be based on maximizing the log-likelihood of the model, without penalizing for structure complexity since the number of parameters of any recursive n-choose-k model is bounded by a polynomial (i.e., regardless of its underlying vtree).

Our approach relies on the following result, which shows that the log-likelihood of a recursive n-choose-k model M, denoted $\mathcal{LL}(M \mid \mathcal{D})$, decomposes over vtree nodes.

Theorem 2 *Consider a recursive n-choose-k model M over variables $\mathbf{X}$ and a corresponding dataset $\mathcal{D}$. Let v be an internal vtree node of this model. We then have*

$$\begin{aligned}\mathcal{LL}(M \mid \mathcal{D}) &= -N \cdot \sum_v H(\mathbf{X}_{v_1} \mid \mathbf{X}_v) \\ &= -N \cdot \sum_v H(\mathbf{X}_{v_2} \mid \mathbf{X}_v)\end{aligned}$$

where $H(\mathbf{X}_{v_1} | \mathbf{X}_v)$ is the (empirical) conditional entropy of the cardinality of $\mathbf{X}_{v_1}$ given the cardinality of $\mathbf{X}_v$:

$$-\sum_{\theta_{v,i}(i_1,i_2)} Pr_{\mathcal{D}}(\mathbf{X}_{v_1}:i_1, \mathbf{X}_v:i) \cdot \log Pr_{\mathcal{D}}(\mathbf{X}_{v_1}:i_1 \mid \mathbf{X}_v:i).$$

Theorem 2 suggests a greedy heuristic for selecting a vtree. We start with n vtrees, each over a single variable $X \in \mathbf{X}$. We greedily pair two vtrees v_a and v_b (i.e., make them children of a new vtree node v) if they have the lowest conditional entropy $H(\mathbf{X}_{v_a} \mid \mathbf{X}_v) = H(\mathbf{X}_{v_b} \mid \mathbf{X}_v)$ over all pairs v_a and v_b. We iterate until we have a single vtree over all variables $\mathbf{X}$.

Example 6. Figure 3 highlights the first few iterations of our vtree learning algorithm, using our course selection example. Initially, at iteration $i = 0$, we have six vtrees, one for each of the six courses. Over all pairs of vtrees, say we obtain the lowest conditional entropy with $H(\{\mathtt{LG}\}|\{\mathtt{LG}, \mathtt{ML}\})$. At iteration $i = 1$, we pair the

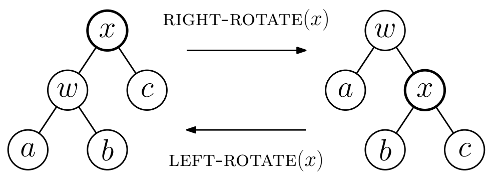

Figure 4: Rotating a vtree node x right and left. Nodes a, b, and c may represent leaves or subtrees.

vtrees of ML and LG to form a new vtree over both. Over all pairs of vtrees, say we now obtain the lowest conditional entropy with $H(\{\texttt{CX}\} | \{\texttt{CX}, \texttt{CP}\})$. At iteration $i = 2$, we again pair the corresponding vtrees to form a new one. We repeat, until we obtain a single vtree. $\square$

Our structure learning algorithm improves the quality of this vtree using local search (simulated annealing in particular). To navigate the full space of vtrees, it suffices to have (left and right) tree rotation operators, and an operator to swap the labels of two vtree leaves.[1] Figure 4 highlights the rotation operator for vtrees. In our experiments, we used simulated annealing with the above operators to navigate the search space of vtrees, using the greedily found vtree we just described as the initial vtree.

Our local search algorithm defines two vtrees as neighbors if one can be obtained from the other by performing a left/right rotation of an internal vtree node, or by swapping the variables of two leaf vtree nodes. We used an exponential cooling schedule for simulated annealing. That is, during each iteration of the algorithm, we first select a neighbor at random. If it has a better log-likelihood score, we move to that neighbor. Otherwise, we move to that neighbor with probability $\exp\{\frac{1}{T_i}\Delta\mathcal{LL}(M \mid \mathcal{D})\}$, where $\Delta\mathcal{LL}(M \mid \mathcal{D})$ is the difference in log-likelihood, and T_i is the temperature at the current iteration. We stop when the temperature drops to a preset threshold.[2] We do not use restarts (empirically, our greedy heuristic appears to obtain a reasonably good initial vtree).

Finally, we remark on the simplicity of our structure learning algorithm, compared to other tractable representations such as the arithmetic circuit (AC) and their sum-product network (SPN) variant (Choi & Darwiche, 2017). In Section 6, we discuss how our recursive n-choose-k model corresponds to a certain type of AC. However, rather than search for ACs (Lowd & Domingos, 2008; Gens & Domingos, 2013; Dennis & Ventura, 2015), we only need to search for vtrees (a much simpler space). This is possible due to a property called canonicity, which fixes the structure of the AC once the vtree is fixed (Kisa et al., 2014).

[1](Choi & Darwiche, 2013) used rotation operators and an operation that swapped the children of an internal vtree node in order to navigate the space of vtrees. In our recursive n-choose-k model, the log likelihood is invariant to the swapping operator, hence we chose to swap the labels of leaf nodes.

[2]In our experiments, we have an initial temperature of 5.0, a cooling rate of 0.98, and a temperature threshold of 0.001.

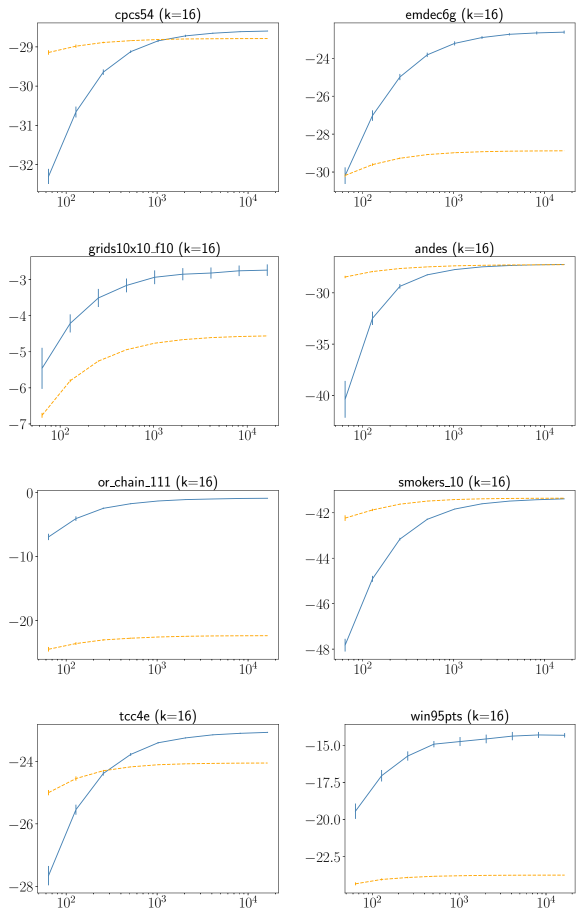

Figure 5: Learning results for cardinality-16: dataset size (x-axis) vs test log likelihood (y-axis). The blue solid lines and orange dashed lines correspond to the recursive and logistic n-choose-k models, respectively.

5 EXPERIMENTS AND CASE STUDIES

We next compare our recursive n-choose-k model with the logistic n-choose-k model of (Swersky et al., 2012), using simulated n-choose-k datasets. We later evaluate these models using real-world datasets from the domains of preference learning and sports analytics.

5.1 SIMULATED DATA

Based on Proposition 1, for a given logistic n-choose-k model, there exists a parameterization of a recursive n-choose-k model that induces the same distribution. However, the logistic n-choose-k model has fewer parameters, as discussed before. Thus, for less data we generally expect the logistic version to be more robust to overfitting, and for greater amounts of data we generally expect our recursive version to ultimately provide a better fit.

The first goal in our experiments is to verify this behavior. We simulated n-choose-k datasets, that are independent of both the logistic and recursive n-choose-k models (so that neither model would be able to fit the data perfectly). In particular, we simulated datasets from Bayesian and Markov networks, but subjected the networks to cardinality-k constraints.[3]

We selected a variety of networks from the literature, over binary variables (the corresponding variable count is given in parentheses): `cpcs54` (54), and `win95pts` (76) are classical diagnostic BNs from the literature; `emdec6g`(168) and `tcc4e` (98) are noisy-or diagnostic BNs from HRL Laboratories; `andes` (223) is a Bayesian network for educational assessment; `grid10x10_f10` (100), `or_chain_111` (200), `smokers_10` (120) are networks taken from previously held UAI competitions. We first considered cardinality-16 datasets. For each, we simulated a testing set of size $2,500$ and independently sampled 20 training sets each of size 2^s for s from 6 (64 examples) to 14 (16,384 examples). We trained n-choose-k models from each training set, and evaluated them using the testing set, in Figure 5. Each point in a plot is an average over 20 training sets.

Our recursive n-choose-k model is depicted with a solid blue line, and the logistic n-choose-k model is depicted with a dashed orange line. There are a few general trends. First, the logistic n-choose-k model more often provides a better fit with smaller amounts of data, but in all but two cases (`smokers_10` and `andes`), our recursive alternative will eventually learn a better model given enough data. In Appendix B, we ran analogous experiments except using cardinality-32 datasets, where we observe that our recursive n-choose-k model tends to perform even better, i.e., it tends to overtake the logistic one with fewer examples. Finally, we note that the variance (plotted using error-bars) of the logistic n-choose-k is smaller (since it has fewer parameters).

Figure 6 highlights the impact of varying k, using data simulated from the `win95pts` network. Here, observe the gain obtained from using the recursive model versus the logistic model, in terms of the test log likelihood, i.e., $\mathcal{LL}_{\text{recursive}}(\mathcal{D}) - \mathcal{LL}_{\text{logistic}}(\mathcal{D})$. Hence, if the gain

[3]We remark that it is non-trivial to simulate a Bayesian network, when we condition on logical constraints. To do so, efficiently, we first compiled a Bayesian network to a PSDD, and then multiplied it with a (uniform) PSDD representing a cardinality-k constraint (Shen et al., 2016). The result is a PSDD, which we can now efficiently sample from.

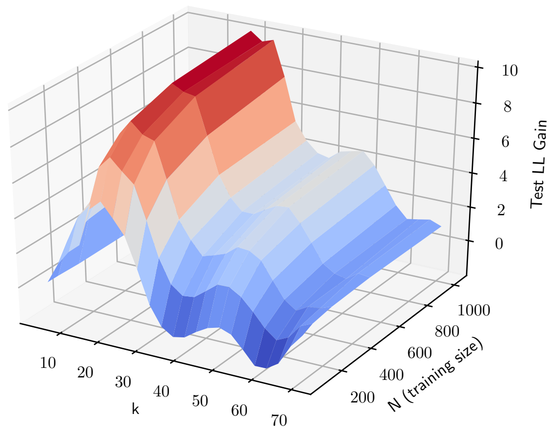

Figure 6: Learning results on the `win95pts` dataset: k vs dataset size vs test log likelihood.

is negative, the logistic model obtained a better likelihood, and if the gain is positive, then our recursive model obtained a better likelihood. Again, as we increase the size of the dataset, our recursive n-choose-k model obtains better likelihoods. As we vary the cardinality of the examples in the data, we see that the performance can vary. Generally, as we increase k from 0 to n, the difference between the models become greater up to a point and then it decreases again. This is expected to an extent since there is an underlying symmetry: a constraint that k values be positive is equivalent to a constraint that $n-k$ values are negative. Hence, for a given n-choose-k model, there is an equivalent n-choose-$(n-k)$ model where the signs have been switched. However, there is not a perfect symmetry in the distribution, since the original distribution generating the data (`win95pts` in this case) does not have this symmetry across cardinalities.

Finally, we remark on the running time of structure learning. On the `win95pts` network, with a 76-choose-32 dataset of size 2^{13}, our structure learning algorithm runs for only 65.02s (on average over 20 runs). On the `andes` network, with a 223-choose-32 dataset of size 2^{13}, it runs for only 367.8s (on average over 20 runs). This is in contrast to other structure learning algorithms for tractable models, such as those based on ACs and their SPNs variant, where learning requires hours for comparably sized learning problems, and even days and weeks for larger scale problems; see, e.g., (Rahman, Kothalkar, & Gogate, 2014) for a discussion. While our recursive n-choose-k model corresponds to a special class of ACs (as we shall discuss in Section 6), it suffices to learn a vtree and not the AC itself.

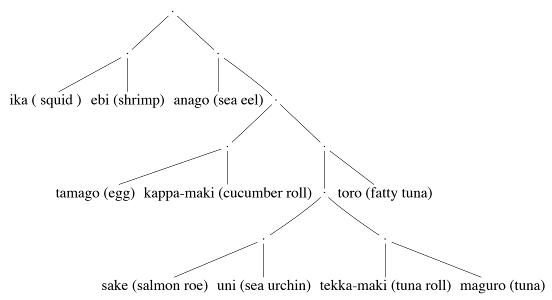

Figure 7: 10-choose-5 model for the `sushi` dataset.

5.2 CASE STUDY: PREFERENCE LEARNING

We consider a case study in preference learning. The `sushi` dataset consists of $5,000$ total rankings of 10 different types of sushi (Kamishima, 2003). From this dataset over total rankings, we can obtain a dataset over top-5 preferred sushi, where we have 10 variables (one for each type of sushi) and a variable is true iff they are in the top-5 of their total ranking; we thus ignore the specific rankings. Note that the resulting dataset is complete, with respect to top-5 rankings. We learned a 10-choose-5 model from this data, using simulated annealing seeded with our conditional entropy heuristic.[4]

Figure 7 highlights the learned vtree structure, which we can view as providing a recursive partitioning to guide a selection of the top-5 preferred sushi, as in Section 3. Going roughly left-to-right, we start with 3 popular non-fish types of sushi: squid, shrimp and sea eel. Next, egg and cucumber roll are relatively not preferred; next, fatty tuna is heavily preferred. Finally, we observe salmon roe and sea urchin (which are both considered acquired tastes) and then tuna roll and tuna. These observations are consistent with previously made observations about the `sushi` dataset; see, e.g., (Lu & Boutilier, 2011; Choi, Van den Broeck, & Darwiche, 2015). In contrast, (Lu & Boutilier, 2011) learned a mixture-of-Mallows model with 6 components, providing 6 different reference rankings (and a dispersion parameter).[5] (Choi et al., 2015) learned a PSDD, but without learning a vtree; a fixed vtree was used based on rankings, which does not reflect any relationships between different types of sushi.

Appendix B compares the recursive and logistic n-choose-k models, where we again observe that the recursive model obtains a better fit when we have more data.

[4]The sushi data was split into a training set of size $3,500$ and a testing set of size $1,500$ as in (Lu & Boutilier, 2011). Our model was learning using just the training set.

[5]The Mallows (1957) model is a probabilistic model for ranking, which assumes a *reference* ranking σ, with other rankings σ' becoming less likely as their distance from σ increases.

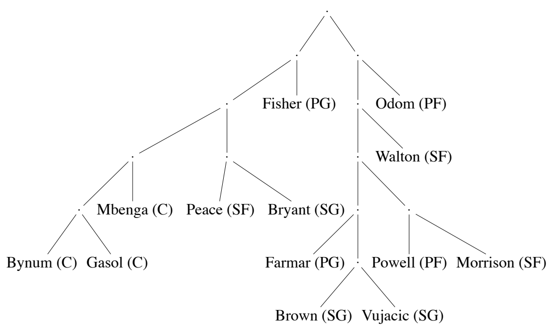

Figure 8: 13-choose-5 model for the 2009-2010 Lakers.

5.3 CASE STUDY: SPORTS ANALYTICS

Team sports, such as basketball and soccer, have an inherent n-choose-k problem, where a coach has to select k out of n players to fulfill different roles on a team. For example, in modern basketball, a team consists of five players who play different roles: there are two guards, who are passers and long-distance shooters; there are two forwards, who generally play closer to the basket; and there is the center, who is normally the tallest player and is the one mostly responsible for gathering rebounds and for contesting shots.

`http://stats.nba.com` provides, for a given season and team, a record of all lineups of five players that played together at the same time, and for how many minutes they played. There are 48 minutes played in a basketball game, and 82 games played during the regular season, for a total $3,936$ minutes. For the 2009-2010 Los Angeles Lakers, that season's NBA champions, we obtained a 13-choose-5 dataset with $39,360$ examples, taking lineups in $\frac{1}{10}$-th minute increments, plus some additional examples due to overtime.

Figure 8 highlights the (vtree) structure that we learned from the full dataset. Again, we can view this structure as providing a recursive partitioning to guide our selection of a five-player NBA lineup, as in Section 3. Starting from the left, and rotating around the root: Andrew Bynum, Pao Gasol and Didier Ilunga-Mbenga are the centers; Metta World Peace, Kobe Bryant, Derek Fisher, and Lamar Odom are the starting non-centers; Luke Walton, Josh Powell and Adam Morrison are the reserve forwards; and Jordan Farmar, Shannon Brown, and Sasha Vujacic are the reserve guards.

Appendix B compares the recursive and logistic n-choose-k models, where we again observe that the recursive model obtains a better fit when we have more data.

6 DISCOVERING THE RECURSIVE n-CHOOSE-k MODEL

We now highlight how the recursive n-choose-k model was *discovered* using the Probabilistic Sentential Decision Diagram (PSDD) (Kisa et al., 2014).

A PSDD allows one to define a probability distribution over a *structured probability space,* which is a subset of the Cartesian product of a set of variables—with each element of this subset corresponding to some object of interest. For example, a structured probability space may correspond to the space of permutations, partial rankings or routes on a map (Choi et al., 2015; Choi, Tavabi, & Darwiche, 2016). PSDDs over structured spaces are interpretable in a precise sense. For certain spaces, including those for subset selection, this interpretability may lead to models whose semantics are so clear that they can be described independently, without the need to invoke the notion of a PSDD in the first place. In such cases, we say that the PSDD has enabled *model discovery*.

Underlying the PSDD is the Sentential Decision Diagram (SDD), which is a class of tractable Boolean circuits (Darwiche, 2011). In this framework, the SDD circuit is used to define the structured probability space (a variable instantiation belongs to the structured space iff the SDD circuit outputs one under that instantiation). Once the structured space is defined by an SDD, a PSDD is used to induce a distribution over that space (the PSDD is basically a parameterized SDD).

Consider the recursive 3-choose-2 model of Figure 2. This model corresponds to an SDD circuit when we replace (1) internal circled nodes with or-gates, (2) paired boxes with and-gates, (3) 1-choose-1 leaf nodes with a positive literal X and 1-choose-0 leaf nodes with a negative literal $\neg X$. The result is an SDD circuit whose satisfying instantiations have exactly two positive literals; that is, the structured probability space for 3-choose-2. See Appendix C for an example of this SDD circuit, and its annotation as a PSDD.

More generally, the following theorem describes SDD circuits whose satisfying instantiations are those with exactly k variables set to true (SDDs are also constructed based on vtrees; see (Darwiche, 2011) for details).

Proposition 2 *Let $f_{v,k}$ be an SDD circuit, for a vtree v with n variables $\mathbf{X}$, whose satisfying instantiations $\mathbf{x}$ set exactly k variables to true. This circuit is equivalent to:*

$$\bigvee_{k_1+k_2=k} f_{v_1,k_1} \wedge f_{v_2,k_2}$$

where $0 \le k_1 \le |\mathbf{X}_{v_1}|$ and $0 \le k_2 \le |\mathbf{X}_{v_2}|$.

Hence, each or-gate of an SDD corresponds to a Boolean formula representing an n-choose-k constraint.

To emphasize the clear semantics of this SDD, consider the number of satisfying instantiations (i.e., model count) that an n-choose-k constraint has: $\binom{n}{k}$. To obtain the model count of an SDD (i.e., the number of satisfying instantiations), we replace each or-gate with a $+$ and each and-gate with a $*$, and all literals with a 1. We then evaluate the circuit bottom-up to evaluate the model count. The model count of the SDD in Figure 2 represents the following computation of $\binom{3}{2}$:

$$\binom{3}{2} = \binom{2}{1}\binom{1}{1} + \binom{2}{2}\binom{1}{0} = 2 \cdot 1 + 1 \cdot 1 = 3.$$

For a more general example, suppose we are given a vtree over a set of n variables $\mathbf{X}$, where the left child of each internal node is a leaf (this is called a right-linear vtree). Computing the model count of an SDD for this vtree, as shown above, yields the well-known recurrence for binomial coefficients:

$$\binom{n}{k} = \binom{1}{0}\binom{n-1}{k} + \binom{1}{1}\binom{n-1}{k-1} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

To obtain a PSDD from an SDD, one assigns a local distribution on the inputs of each or-gate (Kisa et al., 2014). For the SDDs of Proposition 2, these local distributions correspond to the choice distributions of our recursive n-choose-k model; see Appendix C for an example. This observation allowed us to describe this model in a manner independent of the PSDD framework, and hence enabled *model discovery*.

7 CONCLUSION

We proposed in this paper the recursive n-choose-k model for subset selection problems. We also derived a closed-form parameter estimation algorithm for these models, and a simple structure learning algorithm based on greedy and local search. Empirically, we showed how our recursive n-choose-k models can obtain better fits of the data, compared to a previously proposed model. Moreover, we showed how structure search can lead to an intuitive generative model of the subset selection process (based on vtrees). We finally showed how the proposed model was discovered using the PSDD representation for inducing distributions over structured spaces, with the structured space being the set of variable instantiations having a fixed cardinality.

Acknowledgments

This work has been partially supported by NSF grant #IIS-1514253, ONR grant #N00014-15-1-2339 and DARPA XAI grant #N66001-17-2-4032. We thank Tal Friedman, Yitao Liang and Guy Van den Broeck for comments and discussions on this paper.

References

Bekker, J., Davis, J., Choi, A., Darwiche, A., & Van den Broeck, G. (2015). Tractable learning for complex probability queries. In *Advances in Neural Information Processing Systems 28 (NIPS)*.

Chavira, M., & Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence Journal (AIJ)*, *172*(6–7), 772–799.

Choi, A., & Darwiche, A. (2013). Dynamic minimization of sentential decision diagrams. In *Proceedings of the 27th Conference on Artificial Intelligence (AAAI)*.

Choi, A., & Darwiche, A. (2017). On relaxing determinism in arithmetic circuits. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning (ICML)*.

Choi, A., Tavabi, N., & Darwiche, A. (2016). Structured features in naive Bayes classification. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*.

Choi, A., Van den Broeck, G., & Darwiche, A. (2015). Tractable learning for structured probability spaces: A case study in learning preference distributions. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*.

Darwiche, A. (2011). SDD: A new canonical representation of propositional knowledge bases. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 819–826.

Dennis, A. W., & Ventura, D. (2015). Greedy structure search for sum-product networks. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 932–938.

Gens, R., & Domingos, P. M. (2012). Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pp. 3248–3256.

Gens, R., & Domingos, P. M. (2013). Learning the structure of sum-product networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 873–880.

Kamishima, T. (2003). Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 583–588.

Kimmig, A., Van den Broeck, G., & De Raedt, L. (2017). Algebraic model counting. *J. Applied Logic*, *22*, 46–62.

Kisa, D., Van den Broeck, G., Choi, A., & Darwiche, A. (2014). Probabilistic sentential decision diagrams. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Lowd, D., & Domingos, P. M. (2008). Learning arithmetic circuits. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI)*, pp. 383–392.

Lowd, D., & Rooshenas, A. (2013). Learning Markov networks with arithmetic circuits. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 406–414.

Lu, T., & Boutilier, C. (2011). Learning Mallows models with pairwise preferences. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 145–152.

Mallows, C. L. (1957). Non-null ranking models. *Biometrika*, *44*, 114–130.

Meinel, C., & Theobald, T. (1998). *Algorithms and Data Structures in VLSI Design: OBDD — Foundations and Applications*. Springer.

Poon, H., & Domingos, P. M. (2011). Sum-product networks: A new deep architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 337–346.

Rahman, T., Kothalkar, P., & Gogate, V. (2014). Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pp. 630–645.

Shen, Y., Choi, A., & Darwiche, A. (2016). Tractable operations for arithmetic circuits of probabilistic models. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Swersky, K., Tarlow, D., Adams, R. P., Zemel, R. S., & Frey, B. J. (2012). Probabilistic n-choose-k models for classification and ranking. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pp. 3059–3067.

Wegener, I. (2000). *Branching Programs and Binary Decision Diagrams*. SIAM.

A PROOFS

To prove Proposition 1, we reduce the logistic n-choose-k model to a weighted model counting (WMC) problem.

Given a propositional sentence Δ and a set of weights $W(\ell)$ on each literal ℓ, its weighted model count is

$$WMC(\Delta) = \sum_{\mathbf{x} \models \Delta} W(\mathbf{x}) = \sum_{\mathbf{x} \models \Delta} \prod_{\mathbf{x} \models \ell} W(\ell)$$

where the weight of a model $W(\mathbf{x})$ is the product of the weights of its literals $W(\ell)$. For more on weighted model counting see, e.g., (Chavira & Darwiche, 2008; Kimmig, Van den Broeck, & De Raedt, 2017).

A WMC problem induces a distribution over its models:

$$Pr(\mathbf{x}) = \frac{W(\mathbf{x})}{WMC(\Delta)}.$$

If a sentence Δ can be compiled into an SDD, then the SDD can be used to compute its weighted model count. Subsequently, a PSDD can represent the corresponding distribution, as follows.

Lemma 1 *Consider a WMC problem over a propositional sentence Δ with weights $W(\ell)$ on each literal ℓ. Let m be an SDD representing sentence Δ. There is a PSDD with m as its base that induces the same distribution induced by the given WMC problem.*

Proof Given a normalized SDD for Δ, we show how to parameterize it as a PSDD. For an SDD/PSDD node m, let P_m be the distribution induced by the WMC problem on m, and let Q_m be the distribution induced by the PSDD. If m is a terminal node, set $Q_m(\ell) = \eta \cdot W(\ell)$ if ℓ is compatible with the base of m and 0 otherwise, where η is a normalizing constant so that Q_m sums to one. If m is a decision node with elements (p_i, s_i, θ_i), set

$$\theta_i = \frac{WMC(p_i) \cdot WMC(s_i)}{WMC(m)}.$$

We show $P_m(\mathbf{x}) = Q_m(\mathbf{x})$ for all $\mathbf{x}$, by induction. The base case, where m is a terminal node, is immediate. Suppose m is a decision node with elements (p_i, s_i, θ_i) with prime variables $\mathbf{X}$ and sub variables $\mathbf{Y}$, and where $P_{p_i}(\mathbf{x}) = Q_{p_i}(\mathbf{x})$ and $P_{s_i}(\mathbf{y}) = Q_{s_i}(\mathbf{y})$. Given an assignment $\mathbf{xy}$, let the i-th element (p_i, s_i, θ_i) be the one where $\mathbf{x} \models p_i$. We have:

$$\begin{aligned} Q_m(\mathbf{xy}) &= Q_{p_i}(\mathbf{x}) \cdot Q_{s_i}(\mathbf{y}) \cdot \theta_i \\ &= P_{p_i}(\mathbf{x}) \cdot P_{s_i}(\mathbf{y}) \cdot \theta_i \qquad \text{by induction} \\ &= \frac{W(\mathbf{x})}{WMC(p_i)} \cdot \frac{W(\mathbf{y})}{WMC(s_i)} \cdot \frac{WMC(p_i) \cdot WMC(s_i)}{WMC(m)} \\ &= \frac{W(\mathbf{x}) \cdot W(\mathbf{y})}{WMC(m)} = \frac{W(\mathbf{xy})}{WMC(m)} = P_m(\mathbf{xy}). \qquad \square \end{aligned}$$

Proof of Proposition 1 We can represent the logistic n-choose-k model of Equation 1 as a weighted model counting problem problem. First, let Δ be a logical n-choose-k constraint as in Proposition 2. If we use the weights $W(X) = \exp\{\theta_X\}$ and $W(\neg X) = 1$, then the weighted model count gives us the partition function of the logistic n-choose-k model of Equation 1.

Using Proposition 2, we can obtain an SDD for Δ of polynomial size. Using the construction of Lemma 1, we obtain a PSDD that corresponds to a recursive n-choose-k model. This distribution is equivalent to the one induced by the WMC problem, and the one induced by the given logistic n-choose-k model. $\square$

Proof of Theorem 1 Under the recursive n-select-k distribution, the probability $Pr_{w,k}(\mathbf{x})$ is a product of $n-1$ choice parameters. Hence, the log likelihood decomposes as follows:

$$\begin{aligned} \mathcal{LL}(M \mid \mathcal{D}) &= \sum_{a=1}^{N} \log Pr_{w,k}(\mathbf{x}) \\ &= \sum_{v,i} \sum_{\theta_{v,i}(i_1,i_2)} \mathcal{D}\#(\mathbf{X}_{v_1}\!:\!i_1, \mathbf{X}_v\!:\!i) \log \theta_{v,i}(i_1,i_2) \\ &= N \cdot \sum_{v,i} \sum_{\theta_{v,i}(i_1,i_2)} Pr_{\mathcal{D}}(\mathbf{X}_{v_1}\!:\!i_1, \mathbf{X}_v\!:\!i) \log \theta_{v,i}(i_1,i_2) \end{aligned}$$

Note that for each v and i, all of the local choice distributions $\theta_{v,i}$ are independent. Hence it suffices to locally maximize each component:

$$\sum_{\theta_{v,i}(i_1,i_2)} Pr_{\mathcal{D}}(\mathbf{X}_{v_1}\!:\!i_1, \mathbf{X}_v\!:\!i) \log \theta_{v,i}(i_1,i_2)$$

which is basically a cross entropy that is maximized at:

$$\begin{aligned} \theta_{v,i}(i_1,i_2) &= Pr_{\mathcal{D}}(\mathbf{X}_{v_1}\!:\!i_1 \mid \mathbf{X}_v\!:\!i) \\ &= Pr_{\mathcal{D}}(\mathbf{X}_{v_2}\!:\!i_2 \mid \mathbf{X}_v\!:\!i). \qquad \square \end{aligned}$$

Proof of Theorem 2 If we substitute the maximum likelihood estimates of Theorem 1 into the log likelihood of an n-choose-k model we obtain our result.

First, consider the component contributed by a single vtree node v and their choice distribution $\theta_{v,i}$:

$$\begin{aligned} & N \sum_{\theta_{v,i}(i_1,i_2)} Pr_{\mathcal{D}}(\mathbf{X}_{v_1}\!:\!i_1, \mathbf{X}_v\!:\!i) \log \theta_{v,i}(i_1,i_2) \\ &= N \sum_{\theta_{v,i}(i_1,i_2)} Pr_{\mathcal{D}}(\mathbf{X}_{v_1}\!:\!i_1, \mathbf{X}_v\!:\!i) \log Pr_{\mathcal{D}}(\mathbf{X}_{v_1}\!:\!i_1 | \mathbf{X}_v\!:\!i) \\ &= -N \cdot H(\mathbf{X}_{v_1}\!:\!i_1 | \mathbf{X}_v\!:\!i) \end{aligned}$$

which is the conditional entropy distribution. Hence:

$$\mathcal{LL}(M \mid \mathcal{D}) = -N \sum_{v} H(\mathbf{X}_{v_1}\!:\!i_1 | \mathbf{X}_v\!:\!i) \qquad \square$$

Proof of Proposition 2 Consider an n-choose-k constraint $f_{v,k}$ associated with a vtree node v, with children v_1 and v_2 over variables $\mathbf{X}_{v_1}$ and $\mathbf{X}_{v_2}$.

An $\mathbf{X}_{v_1}$-$\mathbf{X}_{v_2}$ decomposition is found by compressing the decomposition:

$$f_{v,k} = \bigvee_{\mathbf{x}_{v_1}} \mathbf{x}_{v_1} \wedge f_{v,k}|\mathbf{x}_{v_1}$$

which is found by disjoining all $\mathbf{x}_{v_1}$ terms that have equivalent terms $f_{v,k}|\mathbf{x}_{v_1}$. For all $\mathbf{x}_{v_1}$ with the same cardinality k_1, the resulting function $f_{v,k}|\mathbf{x}_{v_1}$ is the same. When we disjoin all such $\mathbf{x}_{v_1}$ we obtain the function f_{v_1,k_1}. Further, $f_{v,k}|\mathbf{x}_{v_1} = f_{v_2,k_2}$ for $k_2 = k - k_1$. Hence, the compressed decomposition is:

$$f_{v,k} = \bigvee_{k_1+k_2=k} f_{v_1,k_1} \wedge f_{v_2,k_2}.$$

See also (Meinel & Theobald, 1998; Wegener, 2000) (such as the cardinality-k constraint) for more on symmetric functions on OBDDs. $\square$

B ADDITIONAL EXPERIMENTS

Consider Figure 9, where we ran the same experiments of Figure 5, except where we simulated cardinality-32 datasets instead of cardinality-16 datasets. We observe that our recursive n-choose-k model tends to perform even better here, i.e., they tend to overtake the logistic one with fewer examples.

Consider the preference learning task of Section 5.2, where we considered that `sushi` dataset, which consists of $5,000$ total rankings of 10 different types of sushi (Kamishima, 2003). Consider Figure 10, where we compare our recursive n-choose-k model with the logistic n-choose-k model of (Swersky et al., 2012). First, we split the dataset into a training set (initially, of size 3,500) and a testing set (of size 1,500). Next we simulated training sets of varying sizes, which we used to learn our recursive n-choose-k models, which are then evaluated using the testing set. We used datasets of size 2^s for s from 6 (64 examples) to 11 (2,048 examples), where each was sampled from the original training set, without replacement. Each point of Figure 10 represents an average over 20 simulated training sets. For smaller amounts of data, we see the logistic 10-choose-5 model obtains a better test likelihood. For larger amounts of data, we see our recursive 10-choose-5 model obtains better test likelihoods.

Consider the sports analytics task of Section 5.3, where we considered the 2009-2010 Los Angeles Lakers, that season's NBA champions, and obtained a 13-choose-5

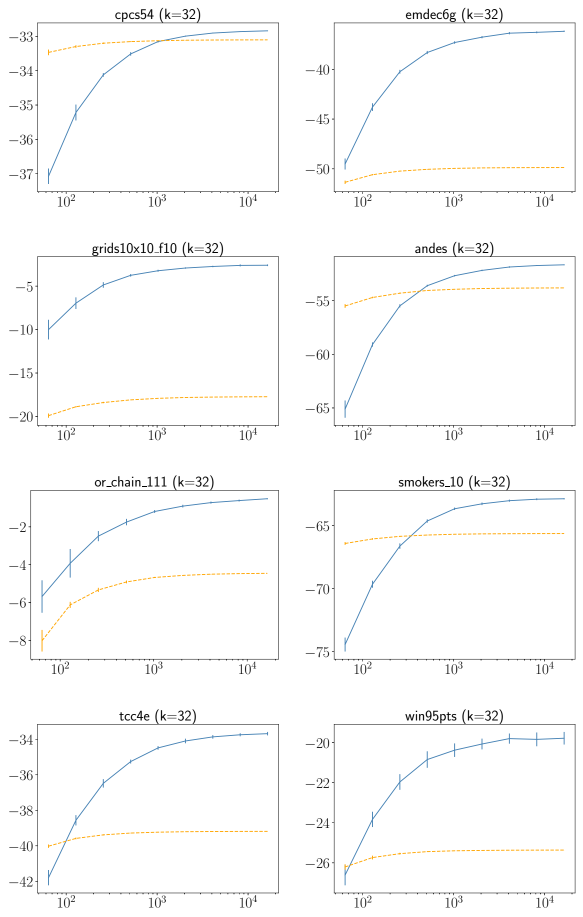

Figure 9: Learning results for cardinality-32: dataset size (x-axis) vs test log likelihood (y-axis). The blue solid lines and orange dashed lines correspond to the recursive and logistic n-choose-k models, respectively.

dataset with $39,360$ examples. In Figure 11, we compared our recursive n-choose-k model with the logistic n-choose-k model. First, we split the dataset into a training set and testing set (the testing set had size 2,500, with the rest going to the training set). Next we simulated training sets of varying sizes, which we used to learn our n-choose-k models, which are then evaluated using the testing set. We used datasets of size 2^s for s from 6 (64 examples) to 14 (16,384 examples), where each was sampled from the original training set, without replacement. Each point of Figure 11 represents an average over 20 simulated training sets. Notably, the logistic model (in orange) does not improve much, even from very small amounts of data. The model that we propose (in blue) provides a better fit, even with a small training set, and is further able to provide increasingly better fits given more data.

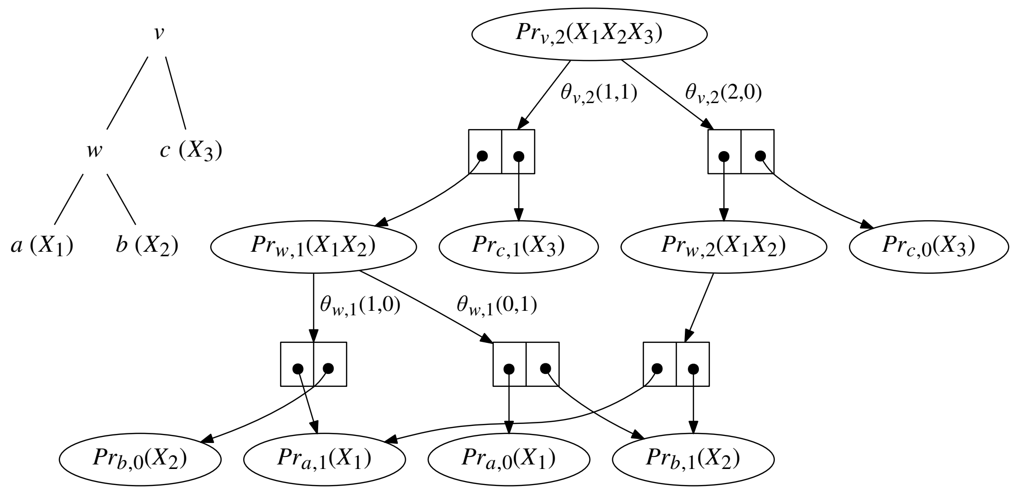

Figure 12: A vtree (upper-left), with a corresponding recursive 3-choose-2 model (right). Leaf vtree nodes are labeled with their variables inside parenthesis. This vtree and model are reproduced from Figure 2.

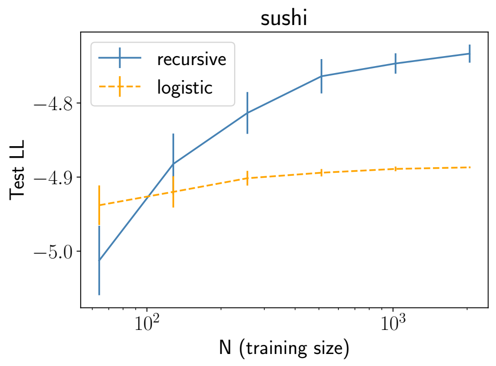

Figure 10: Learning results for the `sushi` dataset.

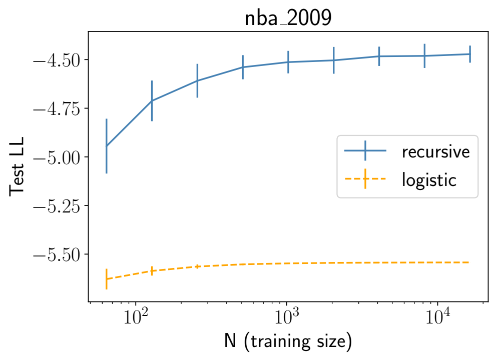

Figure 11: Learning results for the 2009-2010 NBA Champion Los Angeles Lakers.

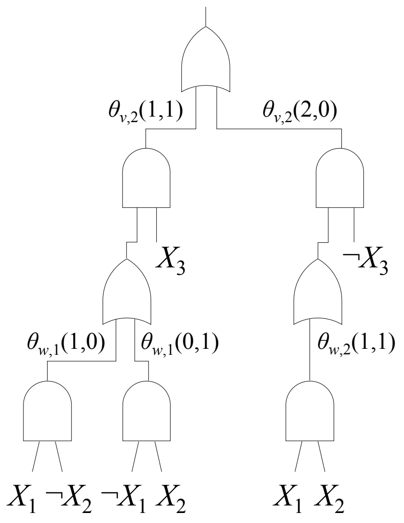

Figure 13: A PSDD corresponding to the vtree and the recursive n-choose-k model of Figure 2 (and Figure 12).

C EXAMPLE PSDD

Figure 13 highlights the SDD/PSDD corresponding to the recursive 3-choose-2 model of Figure 2 using the same vtree. For convenience, we reproduce the vtree and model in Figure 12.

As we highlighted in Section 6, the Boolean circuit of Figure 13 (ignoring the annotated parameters $\theta_{v,k}$) outputs 1 if the circuit input sets exactly 2 out of 3 variables positively, and outputs 0 otherwise. Note that for simplicity, we have omitted inconsistent branches of or-gates that would normally appear in a SDD/PSDD (these branches correspond to instantiations that do not have the required cardinality, and hence, always outputs a 0).

We can obtain an AC of this PSDD by performing two steps: convert each and-gate into a $*$-node, and convert each or-node with children $c_1, \ldots, c_n$ and parameters $\theta_1, \ldots, \theta_n$ into a $+$-node with children $\alpha_1 * c_1, \ldots,$ $\alpha_n * c_n$. Given an instantiation $\mathbf{x}$, the output of the AC is found by setting the inputs to 1/0 according to $\mathbf{x}$ and then evaluating the circuit bottom-up. This output yields the probability $Pr(\mathbf{x})$ of the corresponding recursive 3-choose-2 model.

The properties of SDDs and PSDDs allow certain queries or operations to be performed efficiently, which are otherwise hard on general Boolean and arithmetic circuits. For example, model counting can be performed using SDDs in time that is linear in the size of the SDD (Darwiche, 2011). In PSDDs, queries such as MPE and marginals are similarly tractable (as discussed in Section 3.2). The maximum likelihood parameters of a PSDD can be learned in closed-from from a complete dataset (as in Section 4). Further, one can multiply two PSDDs in polynomial time, which enables incremental learning and inference (Shen et al., 2016).