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 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.

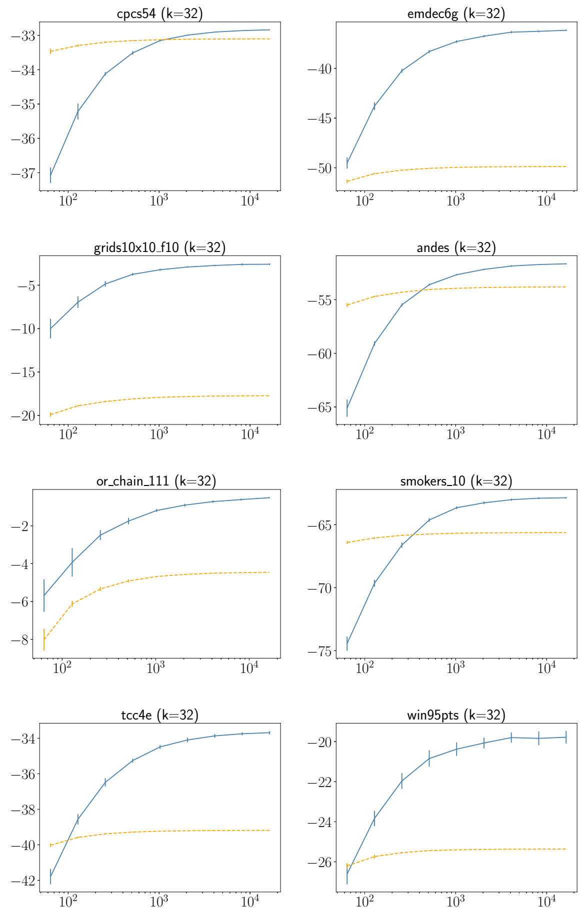

Figure 9: Learning results for cardinality-32: dataset size (x-axis) vs test log likelihood (y-axis). The blue solid lines and orange dashed lines correspond to the recursive and logistic n-choose-k models, respectively.

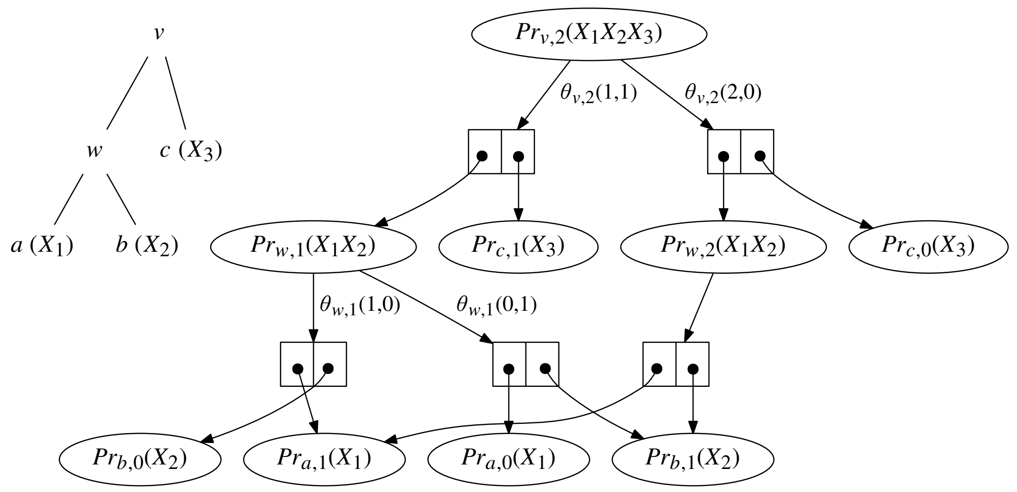

Figure 12: A vtree (upper-left), with a corresponding recursive 3-choose-2 model (right). Leaf vtree nodes are labeled with their variables inside parenthesis. This vtree and model are reproduced from Figure 2.

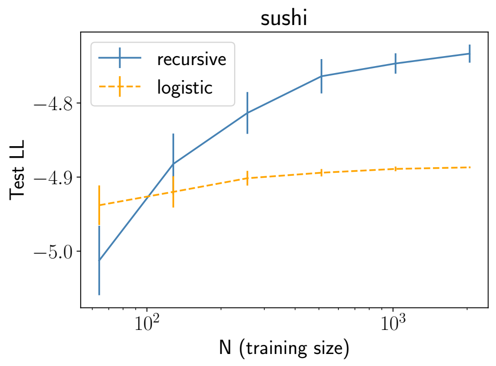

Figure 10: Learning results for the `sushi` dataset.

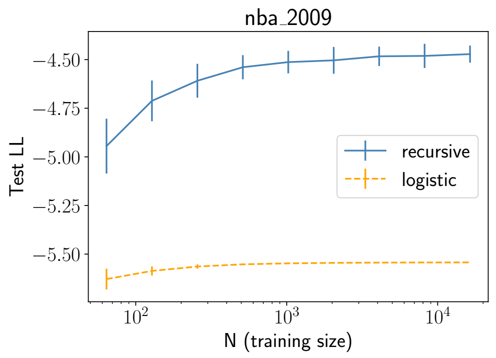

Figure 11: Learning results for the 2009-2010 NBA Champion Los Angeles Lakers.

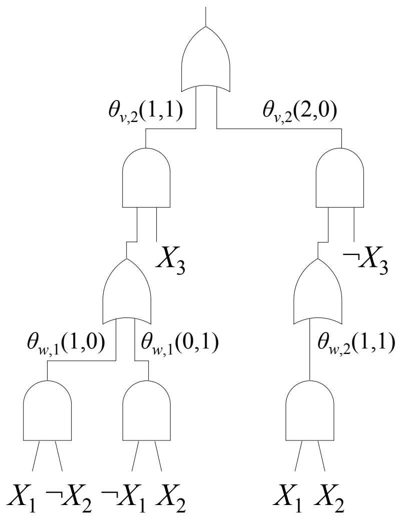

Figure 13: A PSDD corresponding to the vtree and the recursive n-choose-k model of Figure 2 (and Figure 12).

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).