Fast Factorization of Probability Trees and Its Application to Recursive Trees Learning

Andrés Cano, Manuel Gómez-Olmedo,
Cora B. Pérez-Ariza, and Antonio Salmerón

Abstract. We present a fast potential decomposition algorithm that seeks for proportionality in a probability tree. We give a measure that determines the accuracy of a decomposition in case that exact factorization is not possible. This measure can be used to decide the variable with respect to which a tree should be factorized in order to obtain the most accurate decomposed model.

Keywords: Fast factorization, Probability trees, Recursive probability trees.

1 Introduction

The outperformance of trees over other structures in the field of Bayesian networks inference has been analyzed [1, 3]. Trees can be decomposed in order to improve the overall inference process efficiency, by locating the parts of the tree where a concrete operation must be performed. Many algorithms take this into account, being able to work with lists of potentials, as Lazy propagation [6] or Lazy-penniless [4]. Recursive Probability Trees (RPTs) [2] are a generalization of probability trees, and are able to store any possible decomposition of a potential.

There have been previous works on tree-based potential decomposition [7, 8], but the limitations of probability trees make those attempts quite time demanding during the inference process. In this work we present a potential

Andrés Cano, Manuel Gómez-Olmedo, and Cora B. Pérez-Ariza
Dept. of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain
e-mail: acu,mgomez,cora@decsai.ugr.es

Antonio Salmerón
Dept. of Statistics and Applied Mathematics, University of Almería, 04120 Almería, Spain
e-mail: antonio.salmeron@ual.es

decomposition algorithm that quickly divides a probability tree into (approximately) proportional factors that can be stored in a RPT.

2 Recursive Probability Trees

A Recursive Probability Tree [2] is a directed tree with two different kinds of inner nodes: Split and List nodes, and two types of leaf nodes: Value and Potential nodes. A Split node represents a discrete variable. A List node represents a multiplicative factorization by listing all the factors in which a potential is decomposed. It has as many outgoing arcs as factors in the decomposition. A Value node represents a non-negative real number, and finally a Potential node stores a full potential internally represented in whatever representation.

With this structure it is possible to represent context-specific independencies within a probability distribution as well as factorizations (involving the whole potential or parts of it). Sometimes potentials present proportionality relations between several parts of the tree. In the simplest case, a part of the tree can be derived from another just by multiplying by a certain factor. This is the case of a probability tree encoding a joint distribution for X_1 and X_2 (see left part in Fig. 1): the potential for $X_1 = 0$ is proportional to the one corresponding to $X_1 = 1$. The second one can be obtained from the first one multiplying by 4. The right part of the figure shows the recursive probability tree for this potential. This recursive tree needs to store only 6 numbers

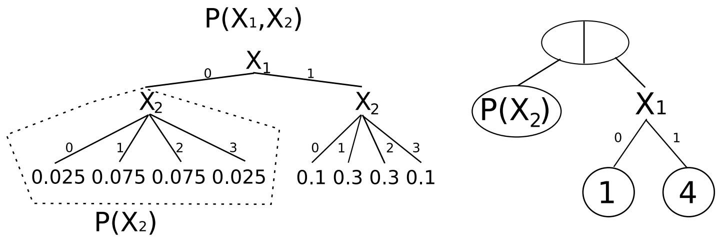

Fig. 1 Potential with proportional values

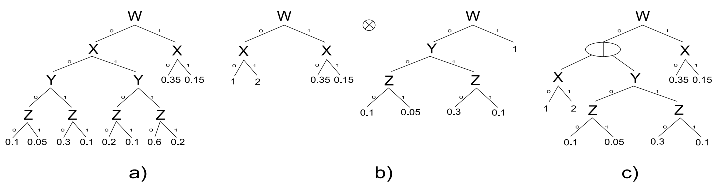

Fig. 2 Decomposition of tree using b) classical decomposition and c) RPTs

instead of 8. For example, an operation involving X_1 need not to work with the left part of the tree.

A RPT can handle this situation and can represent the factorization in a single structure (see Fig. 1). Probability Trees have been used for this kind of decomposition, but usually the factorization must be stored in two independent structures, as in the following example. In the left part of Fig. 2 it can be seen a probability tree encoding a joint distribution for X,Y,Z and W. The result of the factorization taking into account proportional values is shown in the middle part, represented as a multiplication of two probability trees. Finally, this decomposition can be held in just one RPT, as shown in the right part of the figure.

3 Algorithm for Quick Probability Trees Decomposition

When probability distributions are represented with probability trees the efficiency of inference algorithms can be improved by applying tree decomposition, reducing its size and locating the parts that will be affected by a concrete operation. We propose an approximate method for probability trees decomposition, looking for proportional values within itself. The result of the algorithm is a list of factors representing the factorized potential. All these factors can be compactly stored in a RPT.

3.1 Exact Decomposition

The use of probability trees allows to decompose a given tree encoding proportionality into a set of subtrees representing such factorization. However, previous factorization methods search for proportional subtrees located below the variable to delete. Therefore, the performance is highly dependent of the order of the variables in the tree. Let's see an example of this. Consider the probability distribution for variables X, Y and Z represented as the probability tree shown in the left part of Fig. 3. Imagine that X is the next variable to delete during the inference process. Then using classical factorization the most compact decomposition would be the one shown in Fig. 3.

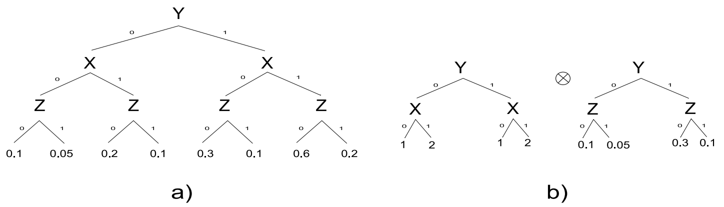

Fig. 3 a) Probability Tree b) Decomposition using classical factorization

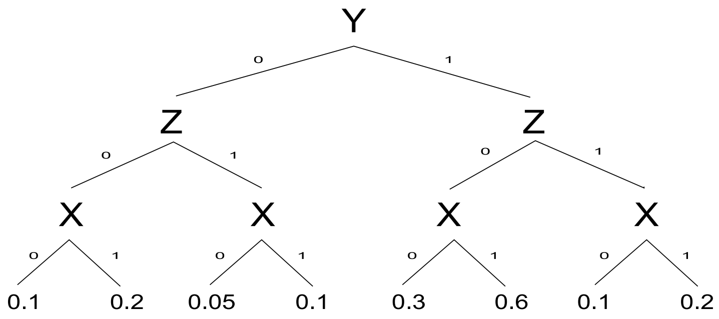

Fig. 4 Probability Tree that can not be factorized by variable X with classical factorization

If the variable to delete is not positioned in the root of the tree, the algorithms that look for factorizations may not be able to find proportionality at all. This happens in the example presented in Fig.4.

We propose an algorithm that finds the most compact factorization of a probability tree for a given variable, independently of the order of the variables in the tree. It returns two probability trees, and we aim to store them as children of a list node within a RPT to benefit from this structure.

```
Input: Probability Tree T and variable to factorize X
Output: two probability trees $\{T_1, T_2\}$
begin
    Let $x_0,\ldots,x_{r-1}$ be the possible values for variable $X$
    Let $(W = w)$ be any configuration for all variables of tree $T$, but $X$
    Let $T^{R(W=w)}$ denote the tree obtained from $T$ by keeping only the
    branches compatible with configuration $(W = w)$
    Let $\alpha_0$, ..., $\alpha_{r-1}$ be the leaves of tree $T^{R(W=w)}$
    for $i \leftarrow 0$ to $r-1$ do
        Let $\beta_i = \frac{\alpha_i}{\alpha_0}$
    end
    Let $T_1$ be a tree with $X$ as only inner node and $\beta_0$, ..., $\beta_{r-1}$ as leaves
    Set $T_2 = T^{R(X=x_0)}$
end
```

Algorithm 1. Factorize(T,X), quick tree factorization algorithm

3.2 Approximate Decomposition

Given a tree, Algorithm 1 finds its most compact decomposition for any variable. However, the exact decomposition for a variable may be impossible just because the tree is not proportional with respect to it. In such case it would be helpful to have a measure about the degree of exact decomposability of a tree for a given variable. Such measure of degree of decomposability could be used, for instance, to establish a threshold to control the accuracy of the decompositions, so that a tree would be allowed to be factorized with respect to a given variable whenever the degree of decomposability surpasses a previously established limit. In this way, it would be possible to obtain an approximate factorization of a tree with a fixed accuracy. It would be achieved by decomposing the tree with respect to any variable that surpass the established limit, and then repeating the same process recursively with the resulting factors, while the limit is surpassed.

If a tree is not exactly decomposable with respect to a given variable, we propose to use the Kullback-Leibler (KL) divergence [5] as a basis to determine how far from exact factorization a given decomposition is. The key result is given in the next theorem, where we give an upper bound of the KL divergence for a given decomposition.

Theorem 1. *Let T be a probability tree to be decomposed with respect to variable X. Let $\mathbf{Y}$ be the set of variables for which T is defined. Let $\mathbf{Z} = \mathbf{Y} \setminus \{X\}$. Let T_1 and T_2 be the output of Algorithm 1 applied to T and X. Then if $D(\cdot,\cdot)$ denotes the KL divergence, it holds that*

$$D(T, T_1 \times T_2) \leq -H(T) - \left(\sum_{\mathbf{y}} t(\mathbf{y})\right)\left(\sum_{x} \log t_1(x) + \sum_{\mathbf{z}} \log t_2(\mathbf{z})\right), \tag{1}$$

where H denotes Shannon's entropy, and t, t_1 and t_2 are the real functions represented by trees T, T_1 and T_2 respectively.

Proof

$$\begin{aligned} D(T, T_1 \times T_2) &= \sum_{\mathbf{y}} t(\mathbf{y}) \log \frac{t(\mathbf{y})}{t_1(x) t_2(\mathbf{z})} = \sum_{x,\mathbf{z}} t(x,\mathbf{z}) \log \frac{t(x,\mathbf{z})}{t_1(x) t_2(\mathbf{z})} \\ &= \sum_{x,\mathbf{z}} t(x,\mathbf{z}) \left(\log t(x,\mathbf{z}) - \log t_1(x) - \log t_2(\mathbf{z})\right) \\ &= \sum_{x,\mathbf{z}} t(x,\mathbf{z}) \log t(x,\mathbf{z}) - \sum_{x,\mathbf{z}} t(x,\mathbf{z}) \log t_1(x) - \sum_{x,\mathbf{z}} t(x,\mathbf{z}) \log t_2(\mathbf{z}) \\ &= -H(T) - \sum_{x}\left(\log t_1(x) \sum_{\mathbf{z}} t(x,\mathbf{z})\right) - \sum_{\mathbf{z}}\left(\log t_2(\mathbf{z}) \sum_{x} t(x,\mathbf{z})\right). \end{aligned}$$

Now, since all the values in T are non-negative real numbers, we find that

$$\begin{aligned} D(T, T_1 \times T_2) &= -H(T) - \sum_{x}\left(\log t_1(x) \sum_{\mathbf{z}} t(x,\mathbf{z})\right) - \sum_{\mathbf{z}}\left(\log t_2(\mathbf{z}) \sum_{x} t(x,\mathbf{z})\right) \\ &\leq -H(T) - \left(\sum_{x,\mathbf{z}} t(x,\mathbf{z})\right)\left(\sum_{x} \log t_1(x) + \sum_{\mathbf{z}} \log t_2(\mathbf{z})\right) \\ &= -H(T) - \left(\sum_{\mathbf{y}} t(\mathbf{y})\right)\left(\sum_{x} \log t_1(x) + \sum_{\mathbf{z}} \log t_2(\mathbf{z})\right). \end{aligned}$$

□

Note that, if a decomposition is exact, then $D(T, T_1 \times T_2)$ is equal to 0, and the further a decomposition is to the exact one, the higher the value of the divergence reaches. Observe also that the upper bound given in Equation (1), actually depends on the specific decomposition through the term $S = \sum_x \log t_1(x) + \sum_{\mathbf{z}} \log t_2(\mathbf{z})$, which suggests that S could be used as a measure of the degree of decomposability of a tree T with respect to variable X. The computation of S is rather fast, as it requires a time linear on the size of T_1 and T_2. But if computing time is a critical issue, S can be again bounded using Jensen's inequality as follows:

$$S = \sum_x \log t_1(x) + \sum_{\mathbf{z}} \log t_2(\mathbf{z}) \leq \log \sum_x t_1(x) + \log \sum_{\mathbf{z}} t_2(\mathbf{z}). \tag{2}$$

Note that the right side of the inequality is faster to compute because the logarithm is applied to the result of the sum, instead of being calculated for all the terms. Hence, we use the reasoning above to formally define the degree of decomposability of a tree with respect to a given variable, called the factorization degree, as follows.

Definition 1 (Factorization degree). *Let T be a probability tree. Let $\mathbf{Y}$ be the set of variables for which T is defined, and $X \subset \mathbf{Y}$. Let $\mathbf{Z} = \mathbf{Y} \setminus \{X\}$. Let T_1 and T_2 be the output of Algorithm 1 applied to T and X. We define the* factorization degree *of T with respect to X as*

$$\mathrm{fd}(T, X) = \log \sum_x t_1(x) + \log \sum_{\mathbf{z}} t_2(\mathbf{z}), \tag{3}$$

where t_1 and t_2 are the real functions represented by T_1 and T_2 respectively.

We have restricted this study to the simplest case, in which the first term of the decomposition contains only one variable. However, the bound given by Theorem 1 can be extended to the case in which T_1 has more than one variable. The complexity of computing the factorization would be higher, but the size of the resulting decomposition would be smaller as well. We leave for a future work the analysis of this case.

4 Experimental Evaluation

In order to illustrate the behavior of the approximate decomposition method introduced in Section 3.2, we have carried out two experiments. The first one is aimed at checking the capability of the factorization degree in Def. 1 for ranking the variables in a probability tree according to the accuracy of the decompositions produced when factorizing with respect to them. The goal of the second experiment is to show that the factorization degree is also able to guide the decomposition of a single probability tree into several factors.

Experiment 1 consists of generating a random tree with 10 binary variables and then computing the factorization degree for each one of the variables and decomposing the tree for the given variable. Then, we measure, using the obtained factorization, the log-likelihood of a test data set sampled from

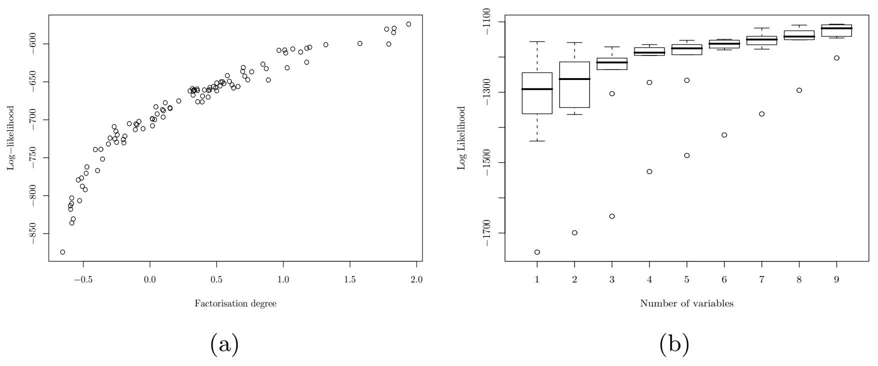

Fig. 5 Results of the experiments carried out.

the original tree. This process is repeated 10 times for different random trees. The obtained results can be seen in Fig. 5.(a).

In experiment 2, we decompose a random tree with 10 binary variables into as many factors as variables it has. Initially, we get the variable with highest factorization degree for the original tree, and decompose it with respect to that variable using Algorithm 1. Then we repeat the same process with the resulting factors, as long as they contain more than one variable, until no more factors can be decomposed. We annotate the log-likelihood of a test data set sampled from the original tree together with the number of variables in the largest remaining factor. Again, the process is repeated 10 times for different random trees. The results are displayed in the box plot in Fig. 5.(b).

The results of experiment 1 support the idea that the factorization degree of the variable with respect to which the decomposition is carried out actually influences the quality of the decomposed model, in the sense that higher factorization degrees result in models with higher likelihood (see Fig. 5.(a)). This fact suggests that the factorization degree could be used as a means for deciding which is the best variable with respect to which a potential should be split. This can be taken into account when designing specific approximate inference algorithms that deal with factorized representations of potentials as, for instance, RPTs. Experiment 2 shows how the factorization degree can also be used to decompose a potential into several factors, by dividing the potential with respect to the variable with higher factorization degree in each step. The decomposition process could be continued until a significant variation in the likelihood of the resulting model happens.

5 Conclusions

In this paper we have introduced a new and fast procedure for factorizing probability trees. An important feature of the proposed algorithm is related to its capability for obtaining optimal decompositions for trees hiding proportionality. Therefore, the resulting decompositions are the most compact ones. We have also shown that the decomposition can be carried out even if the tree does not really contain proportional subtrees, in which case the obtained factorization will be approximate. In order to deal with the degree of approximation of the possible factorizations of a potential, we have introduced a measure called the *factorization degree*, that ranks the variables in the domain of a potential according to the accuracy of the decompositions that they induce. The computation of such measure is fast enough as to be included in any inference algorithm, where computing time is a crucial issue.

The experimental evaluation shows that the methodology developed in this paper can be useful in the design of new inference algorithms that can handle decomposed representations of potentials. Also, it can be used as a tool for learning RPTs or, more precisely, learning parts of a RPT that corresponds to list nodes, and therefore could be a part of a more general learning algorithm for RPTs able to capture all the regularities that a RPT can represent, as for instance, context specific independencies.

Acknowledgements. This work has been supported by the Spanish Ministry of Science and Innovation, under projects TIN2007-67418-C03-02,03, by EFRD (FEDER) funds and the FPI scholarship programme (BES-2008-002049).

References

1. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: Horvitz, E., Jensen, F. (eds.) Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, UAI 1996, Portland, Oregon, pp. 115–123. Morgan Kaufmann, San Francisco (1996)
2. Cano, A., Gómez-Olmedo, M., Moral, S., Pérez-Ariza, C.: Recursive probability trees for Bayesian networks. In: Proceedings of the XIII Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2009, Sevilla, Spain (2009)
3. Cano, A., Moral, S.: Propagación exacta y aproximada con árboles de probabilidad. In: Actas de la VII Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA 1997, Málaga, Spain, pp. 635–644 (1997)
4. Cano, A., Moral, S., Salmerón, A.: Lazy evaluation in Penniless propagation over join trees. Networks 39, 175–185 (2002)
5. Kullback, S., Leibler, R.: On information and sufficiency. Ann. Math. Statist. 22, 76–86 (1951)
6. Madsen, A., Jensen, F.: Lazy propagation: a junction tree inference algorithm based on lazy evaluation. Artificial Intelligence 113, 203–245 (1999)
7. Martínez, I., Moral, S., Rodríguez, C., Salmerón, A.: Factorisation of probability trees and its application to inference in Bayesian networks. In: Gámez, J., Salmerón, A. (eds.) Proceedings of the First European Workshop on Probabilistic Graphical Models, PGM 2002, Cuenca, Spain, pp. 127–134 (2002)
8. Martínez, I., Moral, S., Rodríguez, C., Salmerón, A.: Approximate factorisation of probability trees. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 51–62. Springer, Heidelberg (2005)