Building a Bayesian Factor Tree From Examples

Francesco Palmieri, Gianmarco Romano and Pierluigi Salvo Rossi
Dipartimento di Ingegneria dell'Informazione, Seconda Universita' di Napoli
via Roma 29, 81031 Aversa, Itay
Email: francesco.palmieri@unina2.it
gianmarco.romano@unina2.it
pierluigi.salvorossi@unina2.it

Davide Mattera
Dipartimento di Ingegneria Biomedica,
Elettronica e delle Telecomunicazioni
Universita' di Napoli Federico II
via Claudio 21, 80125 Napoli, Italy
Email: mattera@unina.it

***Abstract*—A criterion based on mutual information among variables is proposed for building a bayesian tree from a finite number of examples. The factor graph, in Forney-style form, can be used as an associative memory that performs probabilistic inference in data fusion applications. The procedure is explained with the aid of a fully-described example.**

I. Introduction

Solving probabilistic problems on graphs via belief propagation [1] has become a very promising paradigm in many fields such as communications, artificial intelligence and digital signal processing [2][3][4]. The bayesian graphical approach shows great potential when we need to integrate smoothly observations and previous knowledge. The idea of "injecting" in a graph our current observations, and "collecting" the response of the system after belief propagation, can be very useful in providing dynamic inference and support to human decision making. The bayesian paradigm presented in the literature under many names, such as Bayesian Networks, Generative Models, Factor Graphs, Markov Random Fields, Associative Memories, etc., reflects a very common intuition about how a "natural" information system should work [5] [6]. Observations and hypotheses about brain cortex functioning [7], seem to confirm that memory is a distributed property of a neural network, and that it is accessed bidirectionally and hierachically.

The literature on the topic is now quite vast. Successful applications of belief propagation to codes [8][9] are well known. Considerable effort is being currently dedicated to improving the paradigm for better message handling and learning. In our work, we emphasize the Factor Graph (FG) formulation [8] [3], that assigns variables to edges, and functions to blocks. Factor Graphs resemble common block-diagrams and seem to provide an easier path to VLSI and FPGA hardware realizations.

In data fusion problems, the challenge is to build systems that can integrate very heterogenous data. We need to learn the mutual relationships among the variables, i.e. we need to "explain" our observations by deriving a "generative model." The model then becomes essentially our distribute memory that, on the basis of observations, can provide associative recall in the form of probabilistic inference.

In this paper we propose an algorithm to build a generative tree on the basis of mutual information measured among the variables. The idea is inspired by the famous Chow and Liu's algorithm [15], which is based on second-order information and builds a tree across a set of random variables. In our approach instead, we build a hierarchical tree by defining new sets of compound variables and use higher-order mutual information. The idea is presented in reference to a fully-blown example where we assume that we have to build the hierarchy solely from a small set of examples. The tree becomes our previous knowledge about the problem and any observation, or partial inference about the variables, is injected in the factor graph that responds with a smooth comparison to its stored memory after message propagation.

A. Building the information tree

A set of random variables $(X_1, X_2, ..., X_N)$ that belong to different finite alphabets $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_N$, are fully characterized by the their joint probability mass function $p(X_1, X_2, ..., X_N)$. All the mutual interactions among the variables, that may be of very different kinds (labels, bits, attributes, etc.), is contained in the structure of p which can be very complicated and unknown. The function p may represent the structure of a phenomenon, a specific generative model, such as a code, or a rule that ties the variables together.

Note that the information globally exchanged among N variables is described by the mutual information

$$I(X_1; X_2; ...; X_N) = \sum_{i=1}^{N} H(X_i) - H(X_1 X_2 ... X_N), \quad (1)$$

where H denotes the entropy [11]. Clearly, if the variables are independent, the mutual information is null and there is no structure because p is simply the product of N marginal densities. In practical cases this is not the case and the dependence is distributed among the variables in structured, often unknown, form. When the dependence is described with a graph, via probability propagation (Bayesian reasoning), we can use our knowledge by computing a posteriori distributions [1]. Common graph topologies are chains, multiple chains, trees, etc. When the graph has loops, inference can become

978-1-4244-6459-3/10/$26.00 ©2010 IEEE

much more complex because standard message propagation does not guarantee convergence to the true marginals. In such cases other approaches must be used, such as the reduction of the graph to a poly-tree (junction tree algorithm), or random sampling (see [2] for references).

Unfortunately, in many practical applications of data fusion, we may have very little knowledge about the system structure and/or its parameters. If we knew the system topology, we would have to solve a parametric problem in searching for the node functions that match our data with a distributed version of the EM-algorithm [6]. Conversely, if we have no idea of what the graph structure should be, we have to build our generative model solely from data, or from our previous knowledge of the problem [19].

We suppose here that we have available only information gathered from a set of N sensors in the form of a finite number of examples $(x_1[i], x_2[i], ..., x_N[i])$, $i = 1, ..., \ell$. Our objective is to "learn" the "generative model" for this set, or equivalently "store" this information in a graph, for further use in probabilistic inference.

Algorithms for growing graphs from data have been proposed in the literature [16], the most famous being Chow and Liu's algorithm [15] that approximates with a tree the joint density function using second order mutual information. Other methods, such as Kutato and K2 [18], starting from independent nodes, add progressively new branches among the variables, following a global entropy optimization. Mixtures of trees have also been proposed in [17], and trees based on greedy searches have been demonstrated in [20]. Also, *module networks*, proposed by Segal et al. [21], are based on information trees with homogeneous groups. The interest in building bayesian trees is clearly in the absence of loops, for which message propagation provides exact marginalization.

In this paper we take the direct approach of building a tree, by grouping variables in a hierarchical order according to their high-order mutual dependence. Often, measuring second-order dependence may not be sufficient (as shown in the example). The hierarchy, in forming new compound variables, can exploit dependence at different tree levels, as it is progressively discovered.

Note that the possibility of inferring on variable, say Y_i, from some knowledge on Y_j, $j \neq i$, depends on how small $H(Y_i|Y_j)$ is, compared to $H(Y_i)$, i.e. on how large $I(Y_i; Y_j)$ is. More in general, to infer on a set of variables from other variables, the mutual information between the two sets is the relevant parameter. Therefore, in building a system that can function as an inference machine, we have to account for the distribution of the mutual information: *dependent variables should be bound together in order to exploit their mutual dependence*.

First note that if the set $X = (X_1, X_2, ..., X_N)$ is split into two subsets

$$\begin{array}{c} X^a = (X_1^a, X_2^a, ..., X_{Na}^a),\ X^b = (X_1^b, X_2^b, ..., X_{Nb}^b), \\ X^a \cap X^b = \emptyset, \quad X^a \cup X^b = X, \quad N_a + N_b = N, \end{array} \tag{2}$$

the mutual information can be decomposed as

$$I(X_1; X_2; ...; X_N) = I(X^a) + I(X^b) + I(X^a; X^b), \tag{3}$$

where

$$I(X^a) = I(X_1^a; X_2^a; ...; X_{Na}^a),\ I(X^b) = I(X_1^b; X_2^b; ...; X_{Nb}^b),$$

are the mutual informations among the elements of each set (*intra*), and $I(X^a; X^b) = I(X_1^a X_2^a ... X_{Na}^a; X_1^b X_2^b ... X_{Nb}^b)$ is the mutual information between the two compound variables X^a and X^b (*inter*). More in general, if X is partionend into n subsets,

$$\begin{array}{ll} X^1 = (X_1^1, ..., X_{N1}^1),\ X^2 = (X_1^2, ..., X_{N2}^2), ... & \\ ...,\ X^n = (X_1^n, ..., X_{Nn}^n), & \sum_{i=1}^n N_i = N, \end{array} \tag{4}$$

the mutual information is decomposed as

$$I(X_1; X_2; ...; X_N) = \sum_{i=1}^{n} I(X^i) + I(X^1; X^2; ...; X^n). \tag{5}$$

Figure 1 shows an example where six variables are grouped hierarchically with the mutual information of each group (intra) shown on each node. The residual information among groups (inter), is shown on the right-end side. Proceeding towards the tree root the residual decreases to zero, because all the information is compounded in the root variable. The most efficient distribution of the information for inference from/to the variables $X_1, ..., X_6$, is obtained when dependent variables are tied together and the residual inter-group information is reduced as much as possible before reaching the tree root. Message propagation in the tree during inference reaches convergence more quickly if the variables that need to exchange information are closely connected. A null residual at a level before the root would mean that at that level the group variables are independent. The search for independent groups in the hierarchy shares some resemblances with the Independent Component Analysis (ICA) paradigm [12] (we maintain that the search for independent objects, or information primitives, should be the objective of any information system).

Note that building a hierachical tree allows for merging of variables also at higher levels (just like variable X_6 in Figure 1). The idea is that a variable should enter the tree at the level where it finds the closest dependence with another node variable.

Given a set of examples mutual information can be easily computed using empirical probability mass functions. Consequently, tree construction proceeds as follows:

(a) Compute the global mutual information $I(X)$ that has to be distributed in the tree; (b) Search for second-order dependence by generating all the partitions with subsets up to order two and measure the residual. If second-order dependence is found, choose the partition that gives the lowest residual. If no second order dependence is found, or the residual is too close to $I(X)$, consider higher orders until appreciable dependence is found. In configurations with equal residuals, choose the partition that has the smallest connectivity; (c) Once the first layer has been formed, repeat the algorithm

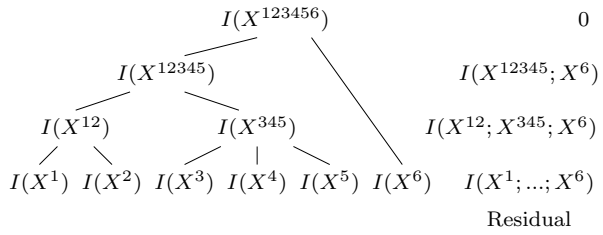

Fig. 1. The information tree from grouping

X_1	X_2	X_3	X_4	X_5	X_6
Y	c	1	*	0	1
Y	b	1	*	0	0
N	a	0	*	0	0
N	b	1	1	0	1
N	c	0	1	1	1
N	d	1	*	1	0
Y	d	1	1	1	1
Y	c	0	*	1	0
N	b	0	1	1	0
N	d	0	*	0	1
Y	a	1	1	1	0
Y	b	0	*	1	1
N	a	1	*	1	1
N	c	1	1	0	0
Y	d	0	1	0	0
Y	a	0	1	0	1

$\mathcal{X}_1 = \{\mathrm{N}, \mathrm{Y}\};$
$\mathcal{X}_2 = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}\};$
$\mathcal{X}_3 = \{0, 1\};$
$\mathcal{X}_4 = \{*, 1\};$
$\mathcal{X}_5 = \{0, 1\};$
$\mathcal{X}_6 = \{0, 1\};$

Fig. 2. The list of examples and the six alphabets.

on the new compound variables and iterate until the tree root is reached with no residual.

Unfortunately, the best hierarchical grouping for a set of variables is not necessarily unique and the search complexity grows quickly with n. However, even adopting suboptimal groupings, the resulting trees are more efficient in message propagation with respect to keeping all the examples lumped into a single compound variable.

In this paper we report the detailed analysis of an example to explain the procedure and demostrate the associative capabilities of the tree. Full computational complexity analysis will be reported elsewhere.

B. Example

Suppose we have available the list of 16 examples shown in Figure 2 for the six variables $X_1, ..., X_6$. We do not know about their mutual relationships, and everything has to be inferred from the available samples (our list of examples is actually obtained from a simple rule, revealed in Appendix A, but we suppose that we do not know that). The variables belong to different alphabets and by counting their occurrences, the marginal entropies (in bits) are $H(X_1) = H(X_3) = H(X_4) = H(X_5) = H(X_6) = 1$, $H(X_2) = 2$. Hence, no first-order information is contained in the examples. Pairs, such as (X_1X_2), (X_2, X_3), (X_5X_6), do not show much either, because (simply by counting occurrences and computing $H(X_1X_2)$, $H(X_3X_4)$, $H(X_5X_6)$) $I(X_1; X_2) = I(X_3; X_4) = I(X_5; X_6) = 0$; i.e. the variables are pairwise independent. Nothing changes if we use different pair partitions. Even with triplets, such as $(X_1X_2X_3)$, we get $I(X_1; X_2; X_3) = 0$. To find dependence we have to go to the fourth order. Grouping variables $(X_1X_2X_3X_4)$, we get the occurrences shown in Figure 3. Half of the strings have zero probability, the others are uniformly distributed. Therefore $H(X_1X_2X_3X_4) = 4$ and $I(X_1; X_2; X_3; X_4) = \sum_{i=1}^{4} H(X_i) - H(X_1X_2X_3X_4) = 1$. The four variables are then mapped in the tree to the new variable Y, shown in the table of Figure 3. Variable Y is uniformly distributed in the alphabet $\mathcal{Y} = \{1, .., 16\}$ with $H(Y) = 4$ (the choice of the alphabet is totally arbitrary). Further grouping of Y and X_5 can be done in the tree since, as shown in the first table of Figure 4, the two variables show dependence. The table reports only the 16 occurrences out of the 32 pairs of the product set $\mathcal{Y} \times \mathcal{X}_5$, in the order of presentation of the examples. The joint entropy is $H(YX_5) = 4$ and $I(Y; X_5) = H(Y) + H(X_5) - H(YX_5) = 1$. Again, the two variables Y and X_5 can be merged to form a new variable Z with alphabet $\mathcal{Z} = \{1, ..., 16\}$ uniformly distributed with entropy $H(Z) = 4$. The last step in the construction of the tree is to consider the pair (ZX_6), shown in the second table of Figure 4. The two variables are dependent because $I(Z; X_6) = H(Z) + H(X_6) - H(ZX_6) = 1$. The tree root is our final variable R uniformly distributed in $\mathcal{R} = \{1, ..., 16\}$.

X_1	X_2	X_3	X_4	p	Y
N	a	0	*	1/16	1
N	a	0	1	0	-
N	a	1	*	1/16	2
N	a	1	1	0	-
N	b	0	*	0	-
N	b	0	1	1/16	3
N	b	1	*	0	-
N	b	1	1	1/16	4
N	c	0	*	0	-
N	c	0	1	1/16	5
N	c	1	*	0	-
N	c	1	1	1/16	6
N	d	0	*	1/16	7
N	d	0	1	0	-
N	d	1	*	1/16	8
N	d	1	1	0	-
Y	a	0	*	0	-
Y	a	0	1	1/16	9
Y	a	1	*	0	-
Y	a	1	1	1/16	10
Y	b	0	*	1/16	11
Y	b	0	1	0	-
Y	b	1	*	1/16	12
Y	b	1	1	0	-
Y	c	0	*	1/16	13
Y	c	0	1	0	-
Y	c	1	*	1/16	14
Y	c	1	1	0	-
Y	d	0	*	0	-
Y	d	0	1	1/16	15
Y	d	1	*	0	-
Y	d	1	1	1/16	16

Fig. 3.

Figure 6 shows the information tree resulting from our construction. The six variables exchange in total $I(X_1; X_2; X_3; X_4; X_5; X_6) = \sum_{i=1}^{6} H(X_i) - H(X_1X_2X_3X_4X_5X_6) = 7 - 4 = 3$ bits of information. The

Y	X_5	p	Z
14	0	1/16	1
12	0	1/16	2
1	0	1/16	3
4	0	1/16	4
5	1	1/16	5
8	1	1/16	6
16	1	1/16	7
13	1	1/16	8
3	1	1/16	9
7	0	1/16	10
10	1	1/16	11
11	1	1/16	12
2	1	1/16	13
6	0	1/16	14
15	0	1/16	15
9	0	1/16	16

Z	X_6	p	R
1	1	1/16	1
2	0	1/16	2
3	0	1/16	3
4	1	1/16	4
5	1	1/16	5
6	0	1/16	6
7	1	1/16	7
8	0	1/16	8
9	0	1/16	9
10	1	1/16	10
11	0	1/16	11
12	1	1/16	12
13	1	1/16	13
14	0	1/16	14
15	0	1/16	15
16	1	1/16	16

Fig. 4.

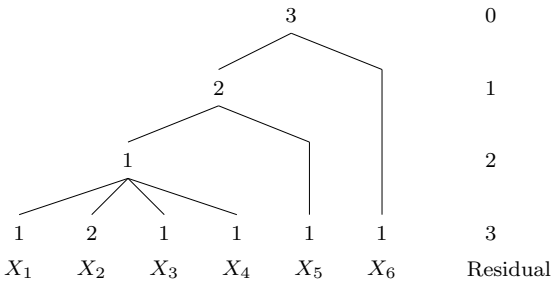

Fig. 6. The information tree for the example.

first group variables $(X_1X_2X_3X_4)$ exchange 1 bit; when X_5 is also added they exchange 2 bits; finally, when all the variables are included we have a total of 3 bits.

Figure 5 shows the complete final tree in the form of a Forney-style factor graph [8][3]. Note the presence of the vertical bars that are the "=" blocks. The variables of the "equal" blocks are split into "equal" variables for easier message propagation.

The factors in the graph are described by the following matrices

$$P(X_1|Y) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, P(X_2|Y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$P(X_3|Y) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, P(X_4|Y) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix},$$

$$P(Y|Z) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$P(X_5|Z) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}, P(Z|R) = I_{16\times 16}, P(X_6|R) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

which are easily derived from the tables of Figures 3 and 4. The prior factor $P_0(R)$ is uniform on the 16 symbols.

1) Pattern recall: The factor tree, can be used as an associative memory: some of the components in X may be unavailable (or erased), or more generally, in a data fusion problem, imperfect knowledge (a probability distribution) about some of them may be available. Via message propagation and the sum-product rule [2] we can get the exact a posteriori probabilities. The information tree also predicts how many bits we can miss for a perfect recall. In the first group we may miss up to 1 bit of information propagating only up to the first tree-level. The second group $(X_1X_2X_3X_4X_5)$ can miss up to two bits of information, but it needs to climb up to the second level. Globally the string can miss up to 3 bits of information to obain a perfect recall.

We have implemented in Matlab complete message propagation in the tree. Probability updates are synchronous (one block per step) and initial values are uniform. We denote for a generic variable X, with bX and fX, backward and forward messages respectively. Suppose that we present to

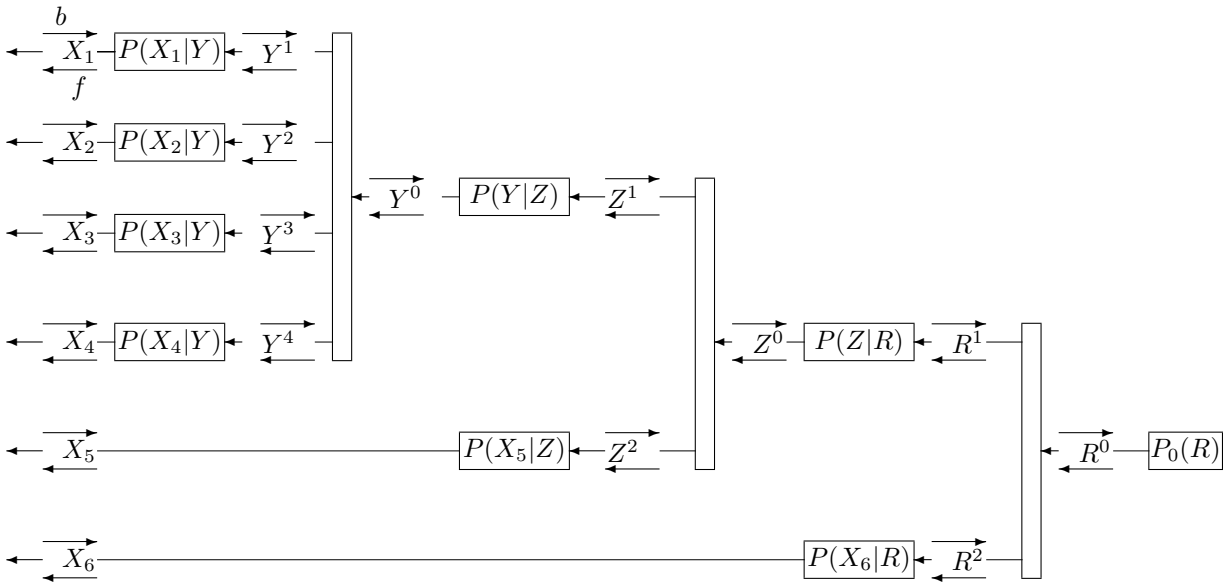

Fig. 5. The tree in the form of a factor graph.

the system the string $(?a1 * 11)$, with the first variable missing. The backward messages injected are $bX_1 = (.5, .5)$, $bX_2 = (1, 0, 0, 0)$, $bX_3 = (0, 1)$, $bX_4 = (1, 0)$, $bX_5 = (0, 1)$, $bX_6 = (0, 1)$. After only three steps we get $fX_1 = (1, 0)$, i.e. $pX_1 = fX_1 \odot bX_1 = (1, 0)$ (perfect recall) where $\odot$ denotes the normalized Hadamard product. [1] The information has been successfully propagated from the neighbours X_2, X_3, X_4 onto X_1 involving only the first layer. No further steps are necessary as predicted by the information tree. If we present the system the string $(Yd??11)$, X_3 and X_4 are missing (two bits). After three steps $fX_3 = (.5, .5)$ and $fX_4 = (0, 1)$. Therefore X_4 is already recalled only using the first layer, but X_3 is still uncertain. After five steps, information from X_5 via the second layer reaches the leaf and $fX_3 = (0, 1)$, giving perfect recall as predicted by the information tree. The maximum number of steps for any pattern that misses three arbitrary bits is seven (tree diameter). If we present the string $(N?11?1)$, we have three bits missing. After three steps, $fX_2 = (0, .5, .5, 0)$ and $fX_5 = (.5, .5)$. Not enough information is yet available, even if a double hypothesis is made on X_2. After five steps, we get $fX_2 = (0, .5, .5, 0)$ and $fX_5 = (1, 0)$, i.e. X_5 is resolved, but X_2 is still uncertain. After seven steps, $fX_2 = (0, 1, 0, 0)$ and the recall is completed.

More interesting for data fusion is the use of the system in "soft mode," assuming that we only have imperfect observation of the six variables. For example, we imagine that we have gathered information about the six variables, but we are not sure about their values and inject our belief in the form of backward messages $bX_1 = (.1, .9)$, $bX_2 = (.25, .25, .25, .25)$, $bX_3 = (.3, .7)$, $bX_4 = (.01, .99)$, $bX_5 = (.9, .1)$, $bX_6 = (.4, .6)$. The system responds after seven steps with $fX_1 = (.6568, .3432)$, $fX_2 = (.4524, .1096, .0807, .3573)$, $fX_3 = (.8136, .1864)$, $fX_4 = (.6280, .3720)$, $fX_5 = (.3432, .6568)$, $fX_6 = (.5, .5)$. After products and normalizations, we get our "improved knowledge" with $pX_1 = (.1754, .8246)$, $pX_2 = (.4524, .1096, .0807, .3573)$, $pX_3 = (.6516, .3484)$, $pX_4 = (.0168, .9832)$, $pX_5 = (.8246, .1754)$, $pX_6 = (.4, .6)$. *The system has responded by shifting our beliefs in a way that accounts for the fixed patterns memorized in tree*. The process can be seen as hard logic (the tree) that interacts with soft knowledge, thanks to the probabilistic framework [5].

2) Recall in the presence of errors: The patterns given to the system to generate the tree are memorized exactly. The graph representation is a useful alternative to simple table memorization because it can provide answers to partial questions using only a subset of variables and it can provide soft probabilistic inference. The memorized $\ell = 16$ patterns are fixed points in the space $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_6$ that has the size $|\mathcal{X}| = 128$ and constitutes the *training set* $\mathcal{T} \subset \mathcal{X}$. However, we would like the system to also function when it is presented with other input configurations that do not belong to $\mathcal{T}$. We need to properly *generalize* to "unseen" patterns. The most appropriate extension is certainly application dependent and reflects our a priori knowledge about the problem. However, we can assume, in the absence of other information, that in the generative model each variable can be spread uniformly and independently on its alphabeth after being generated by the tree. This is equivalent to assuming that we have a *uniform*

[1] We do not provide details about message transformation rules through the blocks (factors), as they are quite standard [3][14]. It is also well known that message transformations and combinations across the factors do not produce valid distributions. In the following, we always present the messages as distributions after normalization.

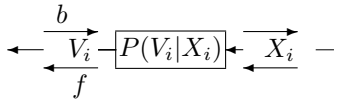

Fig. 7. The noise model

noise model. Figure 7 shows the block to be added at the end of the tree for all $i = 1, ...6$. The conditional probabilities are described by the matrices

$$P(V_i|X_i) = \begin{bmatrix} p_c & 1-p_c \\ 1-p_c & p_c \end{bmatrix}, \quad i = 1, 3, 4, 5, 6,$$
$$P(V_2|X_2) = \begin{bmatrix} p_c & \frac{1-p_c}{3} & \frac{1-p_c}{3} & \frac{1-p_c}{3} \\ \frac{1-p_c}{3} & p_c & \frac{1-p_c}{3} & \frac{1-p_c}{3} \\ \frac{1-p_c}{3} & \frac{1-p_c}{3} & p_c & \frac{1-p_c}{3} \\ \frac{1-p_c}{3} & \frac{1-p_c}{3} & \frac{1-p_c}{3} & p_c \end{bmatrix}, \tag{6}$$

where p_c is the probability of correct generation.

Now our observations correspond to backward messages for $V_i, i = 1, ..., 6$ and the graph diameter is nine. For example, suppose that we observe the string $V = (Nc0 * 00)$. The messages injected in the tree are $bV_1 = (1, 0)$, $bV_2 = (0, 0, 1, 0)$, $bV_3 = (1, 0)$, $bV_4 = (1, 0)$, $bV_5 = (1, 0)$, $bV_6 = (1, 0)$. Assuming $p_c = 0.9$, the system responds after eight steps with $fX_1 = (.2964, .7036)$, $fX_2 = (.8402, .0230, .0230, .1138)$, $fX_3 = (.2964, .7036)$, $fX_4 = (.2964, .7036)$, $fX_5 = (.2964, .7036)$, $bX_6 = (.5, .5)$. After products and normalizations we get $pX_1 = (.7913, .2087)$, $pX_2 = (.5259, .0144, .3885, .0712)$, $pX_3 = (.7913, .2087)$, $pX_4 = (.7913, .2087)$, $pX_5 = (.7913, .2087)$, $bX_6 = (.9, .1)$. Some of the observations have been corrected shifting the probabilities on different symbols.

Mixtures of imperfect or erroneus observations can be presented to the system. The tree on the basis of its training patterns always responds with its opinion (forward message), that joined with the backward message, provides a final inference.

II. Conclusion

In this paper, we have shown, by means of an example, how a fixed set of examples can be used to build an inference tree. Message propagation in the factor graph can be made more efficient with respect to simple table memorization, if we build the tree accounting for the mutual information among the variables in a hierarchical order. The results are preliminary, but show the great potential of this paradigm for building inference machines that can handle incomplete and heterogeneous data.

Appendix

The examples of Figure 2 are obtained hiding in $X_1, ..., X_6$ the (7,4) Hamming code [13] described by the generating equations

$$\begin{aligned} U_1 + U_2 + U_3 + U_5 &= 0 \\ U_2 + U_3 + U_4 + U_6 &= 0 \\ U_1 + U_2 + U_4 + U_7 &= 0 \end{aligned} \tag{7}$$

with $U_i \in \{0, 1\}$. The correspondence with the bits are: $X_1 = U_1$ with $N = 0$ and $Y = 1$; $X_2 = (U_2U_3)$, with $a = (00)$, $b = (01)$, $c = (10)$, $d = (11)$; $X_3 = U_4$; $X_4 = U_5$ with $* = 0$; $X_5 = U_6$; $X_6 = U_7$. The sixteen codewords are listed in a scrambled order. It is well known that the code can be represented also with a Tanner graph (that has loops) and with a trellis (no loops) [3][10].

References

[1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, 2nd ed. San Francisco: Morgan Kaufmann, 1988.

[2] H.A. Loeligher, J. Dauwels, J. Hu, S. Korl, L. Ping and F. Kschischang, "The Factor Graph Approach to Model-Based Signal Processing", *Proceedings of the IEEE*, vol. 95, n.6, June 2007.

[3] H. A. Loeliger, "An Introduction to Factor Graphs," *IEEE Signal Processing Magazine*, pp. 28-41, Jan 2004.

[4] J. Hu, H.A. Loeligher, J. Dauwels and F. Kschischang, "A General Comparison Rule for Lossy Summaries/Messages with Examples from Equalization", *Proceeedings of the 44th Allerton Conference on Communication, Control and Computing*, Monticello, IL, Sept. 27-29, 2006.

[5] E.T. Jaynes, *Probability Theory: The Logic of Science*, Cambridge University Press, 2003.

[6] M.I. Jordan and T. J. Sejnowski, eds.,*Graphical Models: Foundations of Neural Computation*, MIT Press, 2001.

[7] J. Hawkins, *On Intelligence*, Times Books, 2004.

[8] G. D. Forney, Jr., "Codes on graphs: normal realizations," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 520-548, 2001.

[9] Kschischang F.R., B.J. Frey, H.A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. on Information Theory*, Vol. 47, N. 2, pp. 498-519, February 2001.

[10] D. J. C. McKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[12] A. Hyvarinen, J. Karhunen and E. Oja, *Independent Component Analysis*, Wiley, 2001.

[13] S. Benedetto and E. Biglieri, *Principles of Data Transmission with Wireless Applications*, Kluwer Academic Press, 1999.

[14] F. Palmieri, "Notes on Factor Graphs," *New Directions in Neural Networks, IOS Press in the KBIES book series, Proceedings of WIRN 2008, Vietri sul mare*, June 2008.

[15] C. K. Chow and C. N. Liu, "Approximating Discrete Probability Distributions with Dependence Trees," *IEEE Trans. on Information Theory*, Vol. 14, N. 3, May 1968.

[16] J. Cheng, D. A. Bell and W. Liu, "Learning Belief Networks from Data: An Information Theory Based Approach," *Proceedings of the Sixth ACM International Conference on Information and Knowledge Management*, Las Vegas, Nevada, 1997.

[17] M. Meila dn M.I. Jordan, "Learning with Mixtures of Trees," *Journal of Machine Learning Research*, vol. 1, pp. 1-48, October 2000.

[18] E. H. Herskovits and G. F. Cooper, "Kutato: An entropy-driven system for the construction of probabilistic expert systems from databases," *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 5462), Cambridge, MA, 1990.

[19] D. Heckerman, D. Geiger and D. M. Chickering, "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data," *Machine Learning*, Volume 20, Number 3, September 1995.

[20] A. K. Haynes, "Learning Hidden Structure from Data: A Method for Marginalizing Joint Distributions Using Minimum Cross-Correlation Error," Master's Thesis, University of Illinois at Urbana-Champaign, 1997.

[21] E. Segal, D. Pe'er, A. Regev, D. Koller and N. Friedman, "Learning Module Networks," *Journal of Machine Learning Research*, Vol. 6, pp. 557588, 2005.