Graphical Models Project Report
- Belief Propagation -

Charbel-Raphaël Ségerie [*] Clément Bonnet [†]

École Normale Supérieure Paris-Saclay

Abstract

In this work, we study the belief propagation algorithm in two distinct applications. We first apply it to Bayesian networks in a case study [3] to understand the possible uses of the algorithm. Then, we implement belief propagation in the framework of factors graphs, which we used to reproduce a paper [2] to infer the taxonomic tree of a word family from patterns and statistics from Wikipedia abstracts and word embeddings.

1. Introduction

Graphical models are commonly used in probability theory, statistics (especially Bayesian), and machine learning. They are probabilistic models for which a graph expresses the conditional dependence structure between random variables.

The belief propagation algorithm, or sum-product message passing algorithm, calculates the marginal distribution for each unobserved node, conditional on any observed nodes. This algorithm has had successes in numerous applications including free energy approximation and satisfiability. It was first proposed by Judea Pearl in 1982 [4], who formulated it as an exact inference algorithm on trees.

We explain in detail the algorithm and how to implement it in the next sections. We spent a significant amount of time making notations both coherent and simple.

2. Belief Propagation Algorithm

Depending on the type of graph one may deal with, there exist several possible formulations of the belief propagation algorithm. The Bayesian network formulation is presented in section 2.1 whereas the factor graph one is detailed in section 2.2.

2.1. Belief Propagation for Bayesian Networks

Causal Bayesian networks, or simply Bayesian networks, are directed acyclic graphs. The information flows from top to bottom through priors, as well as bottom-up with likelihoods. In such a graph, an edge between two nodes encodes a dependency along with a conditional probability distribution.

To simplify notations, we consider graphs with boolean variables, since one may generalize from booleans to variables taking values in any discrete sets. Capital letters denote nodes while lower-case letters symbolize realizations with bold letters being vectors.

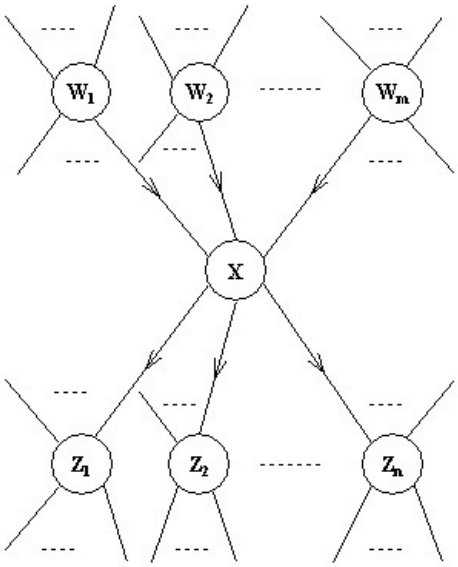

Figure 1: A Bayesian graph centered on a node X.

One such Bayesian network is presented in figure 1, in which we focus on a node X which has m parents and n children.

Propagation Rules

The likelihood is computed in a bottom-up fashion:

$$\lambda(x) = \mathbb{P}(Z_1, \dots, Z_n | X = x) = \prod_{j=1}^{n} \lambda_{Z_j}(x) \tag{1}$$

It must be noted that partial likelihoods $\lambda_{Z_j}(x)$ encode messages in the direction $Z_j \to X$.

We denote $\pi_X(w_k)$ the messages from parents W_k of X flowing to X. Therefore, the prior $\pi(x)$ is propagated to X:

$$\pi(x) = \sum_{\mathbf{w} \in \{0,1\}^m} \mathbb{P}(X = x | \mathbf{w}) \prod_{k=1}^{m} \pi_X(w_k) \tag{2}$$

The belief is then computed as the product between priors and likelihoods. It gathers information about both observations and prior knowledge about unobserved variables.

$$\text{BEL}(x) \propto \lambda(x)\pi(x) \tag{3}$$

[*] charbel-raphael.segerie@hotmail.fr
[†] clement.bonnet16@gmail.com

Update Rules

We now dive into how to update the message in a bottom-up manner, from X to its parents w_i.

$$\lambda_X(w_i) \propto \sum_x \lambda(x) \sum_{\substack{\mathbf{w}' \in \{0,1\}^m \\ w'_i = w_i}} \mathbb{P}(X = x | \mathbf{w}') \prod_{k \neq i} \pi_X(w'_k) \quad (4)$$

This formula allows propagating the likelihood to the node W_i which lives in the upper layer of the network. Among all the scenarios of parents $\mathbf{w}'$, one looks for the ones that are the most frequent (with $\prod \pi$) while explaining well $X = x$ (with its likelihood $\lambda(x)$) and being coherent with the observed children data Z via the likelihood $\lambda(x)$.

Finally, one may also update priors from top to bottom:

$$\pi_{Z_j}(x) \propto \pi(x) \prod_{k \neq j} \lambda_{Z_k}(x) \quad (5)$$

Prior flows from node X to its children Z_j. It is a function of the prior of X and the likelihoods passing from Z_k to X.

Boundary Conditions

- Root nodes: if X is a node with no parents, we set his prior value $\pi(x)$ manually at the beginning.
- Anticipatory nodes: if X is a childless node that has not been instantiated, we set the λ value to be a vector of all 1's.
- Evidence nodes: if evidence $X = x_i$ is obtained, we set the λ value to $(0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 at the i^{th} position.

2.2. Belief Propagation for Factor Graphs

The belief propagation algorithm presented in section 2.1 suffers from a high computational cost. We now look at another type of graph whose architecture may give better performances.

Factor Graph

Factor graphs are bipartite graphs that have two types of nodes:

- Variables, which can be either evidence variables when their value is known, or query variables when it must be predicted.
- Factors, which define the relationships between variables in the graph. They encode one of the factors of the total likelihood of the model.

In a bipartite graph, edges may only connect factors f to variables v and vice-versa. Factors f encode the marginal likelihood of the model with respect to the variable to which they are connected. Hence, the total likelihood of the model is the product of all factors.

Total Likelihood of the Model

The model's total likelihood $p(\mathbf{x})$ may be written as follows:

$$p(\mathbf{x}) = \prod_{f \in F} \mathbf{f}_f(\mathbf{x}_f) \quad (6)$$

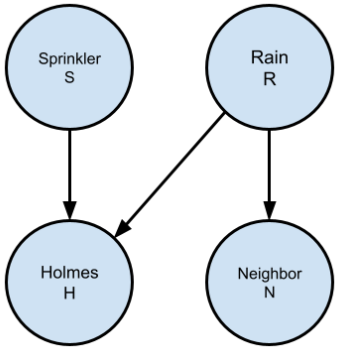

Figure 2: Bayesian network used for our example.

Propagation

Messages from variables to factors are simply computed by multiplying factors that are linked to node v. $\forall x_v \in \text{Dom}(v)$,

$$\mu_{v \to f}(x_v) = \prod_{f^\star \in N(v) \setminus \{f\}} \mu_{f^\star \to v}(x_v) \quad (7)$$

Messages from factors to variables are computed by summing over all counterfactual scenarios. $\forall x_v \in \text{Dom}(v)$,

$$\mu_{f \to v}(x_v) = \sum_{\mathbf{x}'_f : x'_v = x_v} \mathbf{f}_f(\mathbf{x}'_f) \prod_{v^* \in N(f) \setminus \{v\}} \mu_{v^* \to f}(x'_{v^*}) \quad (8)$$

With $\mathbf{x}'_f$ a tuple whose dimension is the number of variables node f is linked to when fixing coordinate x_v.

Marginal Probabilities

Once messages are computed, one may figure out the marginal probability of a variable node by multiplying all adjacent factors.

$$p_{X_v}(x_v) \propto \prod_{f \in N(v)} \mu_{f \to v}(x_v) \quad (9)$$

3. Applications

3.1. Bayesian Networks: Wet Grass Example

We based our Bayesian application on works from [5], [3] and [1]. We adapted and corrected the implementation that we made available on GitHub, whose link is shared in the Acknowledgments section.

We have a situation with a man named Mr. Holmes. One morning when Holmes leaves his house, he realizes that his grass is wet. Is it due to rain (R), or has he forgotten to turn off the sprinkler (S)? *His belief in both events increases.*

- R informs whether it rained or not,
- S tells whether the sprinkler worked or not
- N stands for neighbor, which corresponds the state of the neighbor's grass. $N = 1$ if its grass is wet.
- H stands for Holmes, who is the main character. $H = 1$ if his grass is wet.

Although one may see the causal dependencies between some variables, one may notice that both the sprinkler and the rain are supposed independent as there are no arrows between them.

3.1.1 Computation of Belief Propagation

Let the prior probabilities for R and S be $\pi(R) = (0.2, 0.8)$ and $\pi(S) = (0.1, 0.9)$, both already relatively true because of 3.1.

The different probabilities representing the model of the world are given in the two following tables.

$\mathbb{P}(N\|R)$	$N=0$	$N=1$
$R=0$	0.8	0.2
$R=1$	0	1

$\mathbb{P}(H\|R,S)$	$H=0$	$H=1$
$R=1, S=1$	0	1
$R=0, S=1$	0.1	0.9
$R=1, S=0$	0	1
$R=0, S=0$	1	0

Boundary Conditions

X	BEL(x)	$\pi(x)$	$\lambda(x)$	$\lambda_H(x)$	$\lambda_N(x)$	$\pi_S(x)$	$\pi_R(x)$
R	-	(0.2, 0.8)	-	(0, 1)	(1, 1)	-	-
S	-	(0.1, 0.9)	-	(0, 1)	(1, 1)	-	-
N	-	-	(1, 1)	-	-	(0.1,0.9)	(0.2,0.8)
H	-	-	(0, 1)	-	-	(0.1,0.9)	(0.2,0.8)

In the beginning, $\lambda(H) = (0, 1)$ since Mr Holmes saw that his grass was wet. Root nodes R and S, as well as anticipatory nodes N and H, have their initial values $\lambda_H(x)$, $\lambda_N(x)$, $\pi_S(x)$ and $\pi_R(x)$ filled thanks to boundary conditions.

We then propagate and update the beliefs in the table below.

X	BEL(x)	$\pi(x)$	$\lambda(x)$	$\lambda_H(x)$	$\lambda_N(x)$	$\pi_S(x)$	$\pi_R(x)$
R	(0, 0.8)	(0.2, 0.8)	(0, 1)	(0, 1)	(1, 1)	-	-
S	(0, 0.9)	(0.1, 0.9)	(0, 1)	(0, 1)	(1, 1)	-	-
N	(0.16, 0.84)	(0.16, 0.84)	(1, 1)	-	-	(0.1, 0.9)	(0.2, 0.8)
H	(0, 0.962)	(0.038, 0.962)	(0, 1)	-	-	(0.1, 0.9)	(0.2, 0.8)

We propagated the likelihood using equation (1) in blue. Then we transmitted the prior with equation (2) in green of which we provide an example of such a calculation for $\pi(H = 0) = 0.038$:

$$\begin{aligned}\pi(H=0) &= \mathbb{P}(H=0|R=0,S=0)\pi_H(R=0)\pi_H(S=0)\\ &+ \mathbb{P}(H=0|R=0,S=1)\pi_H(R=0)\pi_H(S=1)\\ &+ \mathbb{P}(H=0|R=1,S=0)\pi_H(R=1)\pi_H(S=0)\\ &+ \mathbb{P}(H=0|R=1,S=1)\pi_H(R=1)\pi_H(S=1)\\ &= 1\times(0.2\times0.1) + 0.1\times(0.2\times0.9)\\ &+ 0\times(0.8\times0.1) + 0\times(0.8\times0.9)\\ &= 0.02 + 0.018 = 0.038\end{aligned}$$

The belief was then computed thanks to equation (3) in magenta. For a better understanding of the belief computation, it is displayed in the table before normalizing it into a probability distribution.

Then, one may compute λ_H, λ_N, π_S and π_R using equations (4) and (5). One of such computations is displayed below to calculate $\lambda_H(R = 1)$.

$$\begin{aligned}\lambda_H(R=1) &= \lambda(H=0)(\mathbb{P}(H=0|R=1,S=0)\pi_H(S=0)\\ &+ \mathbb{P}(H=0|R=1,S=1)\pi_H(S=1))\\ &+ \lambda(H=1)(\mathbb{P}(H=1|R=1,S=0)\pi_H(S=0)\\ &+ \mathbb{P}(H=1|R=1,S=1)\pi_H(S=1))\\ &= 0\times(0\times0.1 + 0\times0.9) + 1\times(1\times0.1 + 1\times0.9) = 1\end{aligned}$$

Since S has no parents, $\pi_H(S)$ is taken to be the prior on S. Therefore, $\pi_H(S = 0) = 0.1$ and $\pi_H(S = 1) = 0.9$.

3.1.2 Simulation

Using our implementation of belief propagation in Bayesian networks, we simulated the computations described in the previous section. We found the following beliefs:

$$\begin{cases}\text{Watson's grass is wet: } 0.865\\ \text{Holmes' grass is wet: } 1\\ \text{It has rained: } 0.832\\ \text{The sprinkler was left on: } 0.917\end{cases}$$

The belief of 1 about Holmes' grass comes from the evidence that the grass was wet. Interestingly, the final belief about whether it rained or not (0.832) has increased compared to the prior on the rain.

Using again the same framework, we found that the belief about rain indeed increased when Holmes could see that both his grass and his neighbor's were wet. The previous calculations are the same, albeit starting with both initial likelihoods $\lambda(H) = (0, 1)$ and $\lambda(N) = (0, 1)$, leading to a final belief about the rain of 0.961 and the sprinkler 0.904. *Hence, this new evidence made Holmes almost sure that it rained.*

This appears to be a powerful framework for belief computations in Bayesian networks. We will now dig into our second application of belief propagation: taxonomy induction with factor graphs.

3.2. Recovering Taxonomy Induction using Belief Propagation on a Factor Graph

We have studied the taxonomy problem by partially reproducing the article [2]. Thus, from a corpus of abstracts on Wikipedia, and a selection of words $(x_1, \ldots, x_n)$, the objective is to discover the membership relations between words and families as in figure 3. We use the Belief propagation algorithm on a factor graph as in figure 4 which encodes the relations between the different pairs of words.

Modeling taxonomic graphs with factor graphs

To build the factor graph as in figure 4, we add a variable node (circle) linked to a factor node (square) for each pair of words (i, j). The variables are boolean whose values are 1 if word x_i is the parent of word x_j. The factors encode the marginal likelihood of one edge $x_i \to w_j$. The factor graph is a visual representation of the probability of a graph $y = (y_{1,1}, \ldots, y_{n,n})$, which is given by:

$$P(y|x) \propto \prod_F \phi_F(y) \tag{10}$$

with F the different factor nodes.

Factors

One may notice that edge factors $E_{i,j}$ (blue squares) only depend on $y_{i,j}$. For each couple (i, j), we extract a set of features $\mho(x_i, x_j)$. Using these features, we compute a dot product with the learned weight vector w. Hence, we get:

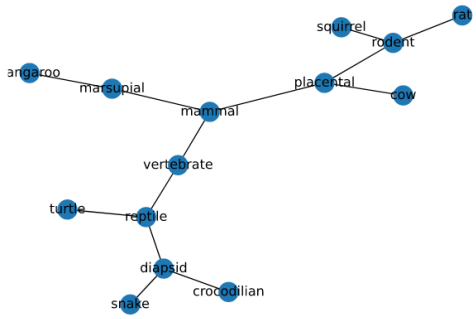

Figure 3: The taxonomic tree we used to calibrate our vector w.

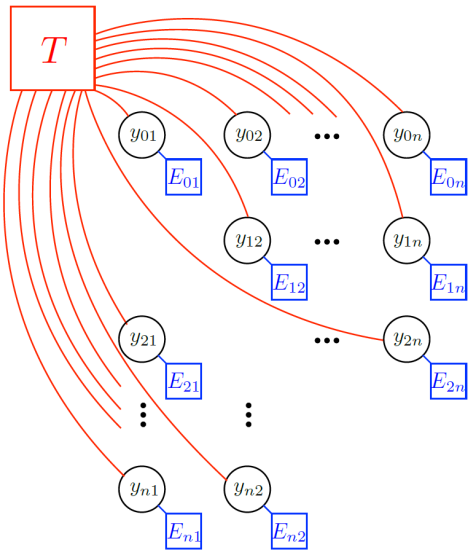

Figure 4: The factor graph used to encode the taxonomic graph.

$$\phi_{E_{i,j}}(y_{i,j}) = \begin{cases} \exp(\mho(x_i, x_j) \cdot w) & \text{if } y_{i,j} = 1 \\ \exp(0) = 1 & \text{if } y_{i,j} = 0 \end{cases} \tag{11}$$

One may also use a tree factor T (red square) to ensure the graph is indeed a taxonomic tree. Its factor value is computed as follows:

$$\phi_T(y_{i,j}) = \begin{cases} 1 & \text{if } y \text{ forms a legal taxonomy tree} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Propagation

The factor graph containing only edge and tree factors is acyclic. In this case, belief propagation is therefore exact. Meaning that after one round of message passing, the beliefs computed are exact.

Embedding

To encode the embedding of the word pair $\mho(x_i, x_j)$, we used Wikipedia API to extract information contained in abstracts as well as Word2vec for a meaningful embedding. From Wikipedia abstracts, we check the occurrence of word i in the abstract of word j. If present, we also compute the distance d_{ij} in the abstract between the two words. Regarding Word2vec, we computed the distance of the two words embeddings $\tilde{x_i} - \tilde{x_j}$. We finally concatenate all these features to obtain a 53-feature vector $f(x_i, x_j)$.

Training

To find the weight vector w that encodes the taxonomic relationship in the graph, we performed a gradient descent computed using a finite difference method on every coordinate.

Results

Gradient descent worked and the algorithm worked as it was able to learn a weight vector w that encodes the taxonomic relationship. However, it found more relationships than necessary as it linked grandparents to grandchildren, e.g., 'cow' to 'mammal'. This may be due to the 'T' node that was not fully implemented and may help with recovering tree shapes instead of graphs.

4. Conclusion

In this work, we have presented the sum-product message-passing algorithm for belief propagation. We have explained in detail its implementation on two kinds of graphs: Bayesian networks and factor graphs.

We studied two applications (one for each type of graph) of belief propagation. The first one enabled us to compute beliefs of unobserved variables that are part of a Bayesian network. The second application allowed us to recover taxonomic trees using a factor graph.

Resources

The code for all experiments is shared here[1]. Slides for the presentation of this work are available there. They may enhance the reader's comprehension and help along with digesting the numerous equations.

References

[1] Wet grass example. http://www.cse.unsw.edu.au/~cs9417ml/Bayes/Pages/PearlExamples.html. Accessed: 29-03-2021. 2

[2] Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1041–1051, 2014. 1, 3

[3] Dawn E Holmes and Lakhmi C Jain. Introduction to bayesian networks. In *Innovations in Bayesian Networks*, pages 1–5. Springer, 2008. 1, 2

[4] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second AAAI Conference on Artificial Intelligence*, AAAI'82, page 133–136. AAAI Press, 1982. 1

[5] Claus Skaanning, Finn V Jensen, and Uffe Kjærulff. Printer troubleshooting using bayesian networks. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 367–380. Springer, 2000. 2

[1] https://github.com/clement-bonnet/belief-propagation