Latent Binary MRF for Online Reconstruction of Large Scale Systems

Victorin Martin * Jean-Marc Lasgouttes † Cyril Furtlehner ‡

Published in *Annals of Mathematics and Artificial Intelligence*, 2015.
The final publication is available at Springer via
http://dx.doi.org/10.1007/s10472-015-9470-x

Abstract

We present a novel method for online inference of real-valued quantities on a large network from very sparse measurements. The target application is a large scale system, like e.g. a traffic network, where a small varying subset of the variables is observed, and predictions about the other variables have to be continuously updated. A key feature of our approach is the modeling of dependencies between the original variables through a latent binary Markov random field. This greatly simplifies both the model selection and its subsequent use. We introduce the mirror belief propagation algorithm, that performs fast inference in such a setting. The offline model estimation relies only on pairwise historical data and its complexity is linear w.r.t. the dataset size. Our method makes no assumptions about the joint and marginal distributions of the variables but is primarily designed with multimodal joint distributions in mind. Numerical experiments demonstrate both the applicability and scalability of the method in practice.

Keywords: latent variables; Markov random field; belief propagation; inference; soft constraints.

1 Introduction

Predicting the behavior of large scale complex stochastic systems is a relevant question in many different situations where a (communication, energy,

*Center for Robotics, Mines ParisTech, PSL Research University, 60 Boulevard Saint-Michel, 75006 Paris, France – victorin.martin@gmail.com

†Inria Paris–Rocquencourt, Domaine de Voluceau, BP. 105, 78153 Le Chesnay CEDEX, France – jean-marc.lasgouttes@inria.fr

‡Inria Saclay–Île-de-France, LRI, Bât 660, Université Paris-Sud, 91405 Orsay CEDEX, France – cyril.furtlehner@inria.fr

transportation, social, economic, etc.) network evolves, for instance with respect to some random demand and limited supply. This remains to a large extent an open and considerable problem, especially for partially observed systems with strong correlations (see e.g. Boyen [5]). Although efficient methods like Kalman and particle filters (see e.g. Doucet et al. [11]) exist, they exhibit limited scalability. The framework presented in this article is based on a probabilistic graphical model (PGM) and avoids modeling the underlying complexity of the physical phenomena. Its two main building blocks are:

- the model itself, built offline from historical data;
- an iterative inference algorithm, that uses the incoming observations to continuously update the prediction about the network state.

Formally, the state of the system is a random vector $\mathbf{X} = (X_i)_{i \in \mathbb{V}}$ of $|\mathbb{V}|$ real valued random variables, attached to nodes $i \in \mathbb{V}$ which take their respective values in the sets $\mathcal{X}_i \subset \mathbb{R}$. The problem at hand is to predict, from sparse local data gathered by moving sensors, the state of the system. Contrary to Eulerian-like approaches, where only a sparse set of fixed locations is observed, the set of nodes to predict is dynamic, due to the sensors' movement. Since only very sparse joint observations are available, purely data-driven methods such as k nearest neighbors (k-NN) cannot be used and one has to resort to building some model of the whole network. Assuming that the sensors behavior is ergodic, all the nodes of the network will eventually be visited an unbounded number of times, making the creation of a statistical model possible. Such an approach would not be possible when sensing happens in fixed places.

The inference problem has to cope with another constraint: prediction on $\mathbf{X}$ must be available in "real-time", which means in practice a few minutes. This constraint implies some design requirements: we need an inference algorithm that scales well and a model that is suitable for use with this inference algorithm. By contrast the estimation of the statistical model via offline processing of historical data can possibly be time consuming.

The example that motivates this work and which will be used for illustration purpose, is road traffic reconstruction from floating car data in an urban context as described in [14, 17]. The density of probe vehicles is supposed to be low, and therefore a model is required to reconstruct the full state of an urban area. The "real-time" constraints for inference are easy to understand in this context, where information is only useful when it is "fresh". This traffic reconstruction application will be described more precisely in Section 7. Swarm robotics, with robots trying to describe collaboratively the state of the area they are exploring could be another possible application.

Building a model of dependency between real-valued variables can be very costly, in terms of statistics, calibration and prediction, if one tries to account for the empirical joint probability distribution of each pair of variables. We choose here another route to model the dependencies of the vector $\mathbf{X}$: to each variable X_i, we attach a latent binary variable σ_i, roughly representing a classification (e.g. good vs. bad state) of the variable X_i. To be able to infer the behavior of these binary variables, given a partial observation of the system, we use a pairwise Markov Random Field (MRF), i.e. an Ising model in statistical physics parlance (see e.g. Baxter [1]). Keep in mind that ultimately we are interested in predicting $\mathbf{X}$.

The problem tackled in this paper can be decomposed into three somewhat related tasks:

- *Encoding*: define the latent variable σ_i in relation to the real valued random variable X_i by specifying $\mathbb{P}(X_i, \sigma_i)$;
- *Model estimation*: construct the joint distribution of the latent variables σ_i, i.e. $\mathbb{P}(\boldsymbol{\sigma})$, in an efficient way in terms of prediction performance;
- *Inference*: once the model is built, insert partial observations of $\mathbf{X}$ into the model to perform the predictions of the unobserved variables.

These three tasks are of course intertwined, but the key component is the inference algorithm. Exact procedures to infer the behavior of the X_i's generally face an exponential complexity, and one has to resort to an approximate procedure. We rely here on Pearl's belief propagation (BP) algorithm [30]—widely used in the Artificial Intelligence and Machine Learning communities [22, 40]—as a basic decoding tool. This choice will be instrumental in the MRF design.

The paper is organized as follows: a formal definition of the model is given in Section 2 along with the main results of the paper. Section 3 is devoted to tackling the encoding task, by finding, for different criteria, optimal mappings between an observation $X = x$ and the parameter of a Bernoulli variable σ. Section 4 focuses on the model estimation task concerning the optimal encoding of X_i's dependencies within the latent space, i.e. the estimation of the underlying MRF. In Section 5, we address inference by introducing mirror BP, a variant of the belief propagation algorithm that imposes belief values equal to $\mathbb{P}(\sigma_i \mid X_i)$ whenever X_i is observed. The implementation and algorithmic complexity aspects are discussed in Section 6. Some experimental results of are presented in Section 7 and the merits of the different encoding/decoding schemes are discussed. Finally, in Section 8, we discuss in particular why binary latent variables have been preferred to other possibilities, like Gaussian variables.

2 Model definition and main results

The model under study is motivated by a moving sensor context where the limited density of sensors prevents acquisition of complete observations of the vector $\mathbf{X}$. We assume that most of the variables are repeatedly observed jointly with some other variables. Pairwise statistics can thus be computed in order to build a model. We retain all pairs of variables $\{(X_i, X_j), (i,j) \in \mathbb{E} \subset \mathbb{V}^2\}$, such that (X_i, X_j) is observed $N_{ij} > n_0$ times, where n_0 is some arbitrary threshold. Note that the set $\mathbb{E}$ could also be limited by a given *a priori* such as knowledge of the neighborhood. We assume all the observations to be independent. For each pair $(i,j) \in \mathbb{E}$, the available observations are summarized as:

$$(X_i, X_j) = (x_i^k, x_j^k) \text{ for } k \in \{1, \dots, N_{ij}\}. \tag{1}$$

From such data, a natural option would be to build a pairwise MRF, in which each interaction pair is justified by relevant statistics of the data. In general it may be advantageous to find an invertible mapping of each variable X_i to some new variable Y_i like e.g. Gaussian variables. The new variables might be more suitable than the input variables, which empirical distributions are not necessarily easy to model. As explained in the introduction, we choose a mapping to binary variables for which efficient pairwise interactions can be obtained.

We assume in our model that the variables $\{X_i\}_{i \in \mathbb{V}}$ are independent, conditionally to the latent state $\boldsymbol{\sigma}$. Along with an invertible mapping, this assumption would be equivalent to model $\mathbf{X}$ by a pairwise MRF. Here it is a design choice: the statistical interactions between variables X_i's are assumed to happen entirely through the latent variables. Since these latent variables are not given in the data, it will be necessary to *construct* them and multiple choices can be meaningful. This amounts to choosing feature functions from $\mathcal{X}_i$ to $\{0,1\}$. The problem at stake, even if it looks similar, is distinct from the inference of the states of a hidden Markov model from noisy observations: in the end the only variables of interest are the X_i's.

Under these assumptions, the joint measure for the variables $\mathbf{X}$ and $\boldsymbol{\sigma}$ factorizes as (Figure 1):

$$\mathbb{P}(\mathbf{X} \leq \mathbf{x}, \boldsymbol{\sigma} = \mathbf{s}) = \mathbb{P}(\boldsymbol{\sigma} = \mathbf{s}) \prod_{i \in \mathbb{V}} \mathbb{P}(X_i \leq x_i \mid \sigma_i = s_i), \tag{2}$$

$$\mathbb{P}(\boldsymbol{\sigma} = \mathbf{s}) = \frac{1}{Z} \prod_{(i,j) \in \mathbb{E}} \psi_{ij}(s_i, s_j) \prod_{i \in \mathbb{V}} \phi_i(s_i), \tag{3}$$

with Z a constant ensuring that $\mathbb{P}$ sums up to 1. The joint distribution of $\mathbf{X}$ is therefore obtained by summing up all latent variables in our PGM and is quite involved. This is not actually a problem, at least for the usage we want to make of this model, i.e. a regression on the variables X_i's. However,

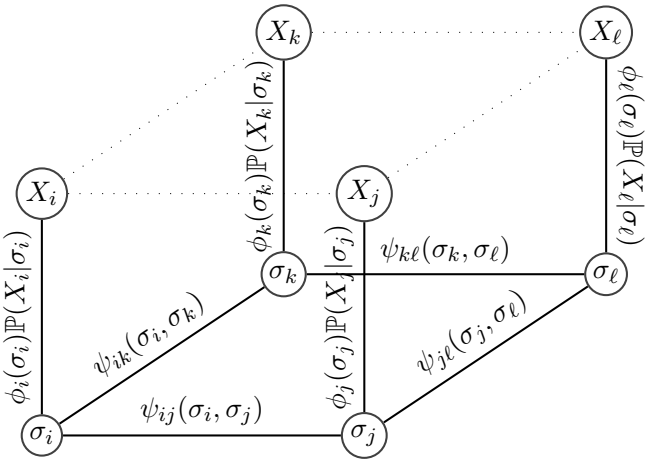


Figure 1: Markov random field $(\mathbf{X}, \boldsymbol{\sigma})$ for $\mathbb{V} = \{i, j, k, \ell\}$. The pairwise PGM of the vector $\mathbf{X}$ to be expected from the data (dotted lines) is approximated through the latent binary variables $\boldsymbol{\sigma}$ (plain lines).

the model estimation has to be performed beforehand and this is usually done via likelihood maximization. This clearly is not tractable because of the presence of the latent variables. We will therefore have to resort to an expectation–maximization (EM) procedure.

Summary of the main results. In this paper, we propose to model the random vector $\mathbf{X}$ by (2) and (3), and we address the three related tasks of the introduction. Let us sum up the solutions which are our main contributions

- *Encoding task*: an answer is provided in Section 3, by finding two optimal mappings, based on different criteria, between an observation $X = x$ and the conditional distribution $\mathbb{P}(\sigma \mid X = x)$. The first one corresponds to a step function that defines σ as a deterministic function of X; the second one is the cumulative distribution function of X for which σ is a random variable conditionally to X. The numerical experiments of Section 7 show that it is more efficient than the step function when the level of correlation increases. Besides, we show with Proposition 1 that such mappings are equivalent to modeling X as a mixture of two stochastically ordered variables. In the traffic context, these two variables can be associated with a free flow and a congested state.

- *Model estimation task*: this is addressed in Section 4; the pairwise marginals $\mathbb{P}(\sigma_i, \sigma_j)$ for any encoding function are estimated using a maximum-likelihood based approach. The task can then be formulated as an Inverse Ising problem. Various methods exists in the literature to approximate the optimal joint distribution $\mathbb{P}(\boldsymbol{\sigma})$; following

e.g. Wainwright [37], this can be approximated directly from these pairwise marginals based on the fact that approximate inference will be performed with BP.

- *Inference task*: our solution, described in Section 5, is a variant of BP named mirror BP that imposes belief values $\mathbb{P}(\sigma_i \mid X_i)$ whenever X_i is observed. This algorithm extends and greatly simplifies the algorithm of Teh and Welling [36]; Sufficient conditions for its convergence are provided in Proposition 5.

3 Latent variables definition

Let X be a real-valued random variable with cumulative distribution function (cdf) $F(x) \stackrel{\text{def}}{=} \mathbb{P}(X \leq x)$. We focus in this section on a way to relate an observable variable X to its latent binary variable σ, i.e. the definition of $\mathbb{P}(\sigma, X)$. Since X is observed we will use its empirical distribution $\mathcal{P}(X)$ and *define* the conditional distribution $\mathbb{P}(\sigma \mid X)$.

Let us emphasize, once more, that σ is not just an unobserved latent random variable which estimation is required. It is a feature that we define in order to tackle the inference on $\mathbf{X}$. A simple way to relate an observation $X = x$ to the latent variable σ is through a mapping Λ, such that $\Lambda(x)$ define the conditional distribution of σ.

3.1 A stochastically ordered mixture

The mapping Λ will be referred to as the encoding function and can depend on the cdf F. σ being a latent variable, it will not be observed directly but, conditionally to an observation $X = x$, we *define* its conditional distribution as:

$$\mathbb{P}(\sigma = 1 \mid X = x) \stackrel{\text{def}}{=} \Lambda(x). \tag{4}$$

For simplicity, we assume that Λ is continuous on right, limited on left (*corlol*) and increasing. Note that Λ could equivalently be decreasing, since choosing the mapping $1-\Lambda$ simply swaps the states 0 and 1 of the variable σ. Moreover, Λ shall increase from 0 to 1, without requiring that $\Lambda(\mathcal{X}) = [0, 1]$, since Λ can be discontinuous. This constraint is expressed as follows:

$$\int_{\mathcal{X}} d\Lambda(X) = 1 \quad \text{and} \quad \inf_{x \in \mathcal{X}} \Lambda(x) = 0.$$

This encoding is part of the following global scheme

$$\begin{array}{ccc} X_i = x_i \in \mathcal{X}_i & \xrightarrow{\Lambda_i} & \mathbb{P}(\sigma_i = 1 \mid X_i = x_i) \in \Lambda_i(\mathcal{X}_i) \\ & & \Big\downarrow \text{ inference algorithm (mBP)} \\ X_j = x_j \in \mathcal{X}_j & \xleftarrow{\Gamma_j} & \mathbb{P}(\sigma_j = 1 \mid X_i = x_i) \in [0, 1] \end{array}$$

which can be stated in plain terms as:

- observations of variables X_i are encoded through the distribution of a latent binary random variable σ_i using the encoding function Λ_i;
- an inference procedure is then performed on these latent variables $\boldsymbol{\sigma}$, in a way that will be described later, to obtain the marginal distributions of the σ_j's;
- the unknown real variables X_j are predicted from the distributions of the σ_j's.

This scheme requires that we associate to Λ an "inverse" mapping $\Gamma : [0,1] \mapsto \mathcal{X}$. Since Λ can be non invertible, the decoding function Γ cannot always be the inverse mapping Λ^{-1}. We will return to the choice of the function Γ in Section 3.3.

To understand the interaction between σ and X, let us define the conditional cdf's:

$$F^0(x) \stackrel{\text{def}}{=} \mathbb{P}(X \leq x \mid \sigma = 0),$$
$$F^1(x) \stackrel{\text{def}}{=} \mathbb{P}(X \leq x \mid \sigma = 1).$$

Bayes' theorem implies

$$\mathbb{P}(\sigma = 1 \mid X = x) = \mathbb{P}(\sigma = 1)\frac{dF^1}{dF}(x),$$

and thus

$$dF^1(x) = \frac{\Lambda(x)}{\mathbb{P}(\sigma = 1)}dF(x). \tag{5}$$

Summing over the values of σ imposes

$$F(x) = \mathbb{P}(\sigma = 1)F^1(x) + \mathbb{P}(\sigma = 0)F^0(x), \tag{6}$$

and the other conditional cdf follows

$$dF^0(x) = \frac{1 - \Lambda(x)}{\mathbb{P}(\sigma = 0)}dF(x).$$

Note that, when $\Lambda = F$, the conditional cdf's of X are actually:

$$F^1(x) = (F(x))^2 = \mathbb{P}(\max(X_1, X_2) \leq x),$$
$$F^0(x) = F(x)(2 - F(x)) = \mathbb{P}(\min(X_1, X_2) \leq x),$$

with X_1 and X_2 two independent copies of X. The cdf's F^0 and F^1 are thus stochastically ordered. This property still holds in the general case, as shown in the following proposition.

Proposition 1. *The choice of an increasing encoding function Λ yields a separation of the random variable X into a mixture of two stochastically ordered variables X^0 and X^1 with distributions dF^0 and dF^1. Indeed, we then have*

$$X \sim \mathbb{1}_{\{\sigma=0\}} X^0 + \mathbb{1}_{\{\sigma=1\}} X^1,$$

where $\sim$ is the equality in term of probability distribution. The stochastic ordering is the following

$$X^0 \preceq X \preceq X^1.$$

Proof. It is sufficient (and necessary) to prove that

$$\forall x \in \mathcal{X}, \quad F^1(x) \leq F(x) \leq F^0(x).$$

Consider first the left inequality ($F^1 \leq F$) and assume that x is such that $\Lambda(x) \leq \mathbb{P}(\sigma = 1)$. We have then:

$$F^1(x) = \int_{-\infty}^{x} dF^1(y) = \int_{-\infty}^{x} \frac{\Lambda(y)}{\mathbb{P}(\sigma = 1)} dF(y) \leq \int_{-\infty}^{x} dF(y) = F(x),$$

because, Λ being increasing, $\Lambda(y) \leq \mathbb{P}(\sigma = 1)$ for all $y \in]-\infty, x]$. In the opposite case, where $\Lambda(x) > \mathbb{P}(\sigma = 1)$, one can write

$$\begin{aligned} F^1(x) &= 1 - \int_{x}^{+\infty} dF^1(y) = 1 - \int_{x}^{+\infty} \frac{\Lambda(y)}{\mathbb{P}(\sigma = 1)} dF(x) \\ &\leq 1 - \int_{x}^{+\infty} dF(y) = F(x), \end{aligned}$$

using again the fact that Λ is increasing. We have proved that $\forall x \in \mathcal{X}$ the inequality $F^1(x) \leq F(x)$ holds. The other inequality $F \leq F^0$ is obtained using (6). ■

Remark 1. Since the encoding function Λ is increasing from 0 to 1, it can be considered as the cdf of some random variable Y,

$$\mathbb{P}(Y \leq x) \stackrel{\text{def}}{=} \Lambda(x),$$

and, assuming that Y and X are independent,

$$\begin{aligned} \mathbb{P}(\sigma = 1) &= \int_{\mathcal{X}} \mathbb{P}(\sigma = 1 \mid X = x) dF(x) = \int_{\mathcal{X}} \Lambda(x) dF(x) = \int_{\mathcal{X}} \mathbb{P}(Y \leq x) dF(x) \\ &= \mathbb{P}(Y \leq X). \end{aligned}$$

The variable σ can therefore be defined as

$$\sigma \stackrel{\text{def}}{=} \mathbb{1}_{\{Y \leq X\}},$$

which means that the variable Y acts as a random threshold separating X-values that correspond to latent states 0 and 1. The stochastic ordering between $(X \mid \sigma = 0)$ and $(X \mid \sigma = 1)$ then appears quite naturally. Note that this interpretation leads to a natural extension to a larger discrete feature space for σ, by simply using multiple thresholds.

3.2 Choosing a good encoding function Λ

Now that the nature of the mapping between X and σ has been described, it remains to find an "optimal" encoding function Λ. It turns out to be difficult to find a single good criterion for this task. In this section, we therefore propose two different approaches, based respectively on mutual information and on entropy.

Mutual information. The idea here is to choose Λ (or equivalently σ), such that the mutual information $I(X, \sigma)$ between variables X and σ is maximized. In other words, a given information about one variable should lead to as much knowledge as possible on the other one.

Proposition 2. *Let* $\mathrm{med}(X)$ *be the median of* X*. The encoding function* Λ^{MI} *which maximizes the mutual information* $I(X, \sigma)$ *between variables* X *and* σ *is the step function*

$$\Lambda^{\mathrm{MI}}(x) \overset{def}{=} \mathbb{1}_{\{x \geq \mathrm{med}(X)\}}. \tag{7}$$

Before turning to the proof of this proposition, let us remark that this definition of the binary variable σ is a natural one, σ being deterministic as a function of X. However, as we shall see in Section 7, it is usually suboptimal for the reconstruction task.

Proof. The function to maximize is

$$\begin{aligned} I(X, \sigma) &= \sum_s \int_{x \in \mathcal{X}} \mathbb{P}(\sigma = s) \log \left(\frac{dF^s(x)}{dF(x)} \right) dF^s(x) \\ &= H(\mathbb{P}(\sigma = 1)) - \int_{x \in \mathcal{X}} H(\Lambda(x)) dF(x), \end{aligned}$$

where $H(p) \overset{\mathrm{def}}{=} -p \log p - (1-p) \log(1-p)$ is the binary entropy function. Among all random variables σ with entropy $H(\mathbb{P}(\sigma = 1))$, the ones which maximize $I(X, \sigma)$ should minimize the second term in the preceding equation. These are deterministic functions of X, or equivalently the ones for which Λ is an indicator function. Since we limit ourselves to the *corlol* class, we get $\Lambda = \mathbb{1}_{[a, +\infty[}$ for some $a \in \mathcal{X}$. It remains to maximize the entropy of the variable σ, which leads to $P(\sigma = 1) = 1/2$ and $a = \mathrm{med}(X)$. ■

Max-entropy principle. Another possibility is, in order to maximize the information contained in the latent variable σ, to maximize the entropy of $U = \Lambda(X)$. This variable U is indeed the data that will be used to build the PGM over the latent variables (see Section 4). We assume here that the variable X admits a probability density function (pdf). We add to the few constraints detailed in the previous section that Λ is a bijection

between $\mathcal{X}$ and $[0,1]$. When dealing with continuous random variables, the entropy only makes sense relatively to some measure (see Jaynes [21, pp. 374-375]). Following Jaynes' [20] arguments, since U is the parameter of a binary variable with both outcomes possible, having no other prior knowledge leads us to the uniform measure as reference.

Proposition 3. *Let X be a random variable which admits a pdf. The (increasing) invertible function which maximizes the entropy of $U = \Lambda(X)$, taken relatively to the uniform measure, is the cumulative distribution function F of the variable X.*

Proof. The variable U with maximal entropy has a uniform pdf $h_\Lambda(u) = \mathbb{1}_{[0,1]}(u)$. The encoding function Λ such that $\Lambda(X)$ is a uniform variable on $[0,1]$ is the cdf of X. This concludes the proof. ■

Let us quickly sum up the choices we proposed for the encoding function:

- the indicator function Λ^{MI} is a deterministic encoding: σ indicates the position of X w.r.t. its median;
- the cdf F of the variable X corresponds to the less discriminating choice about the encoded data distribution.

We will see that these two encoding functions have very distinct properties: Λ^{MI} is much more conservative than F but is rather adapted to modeling precisely the joint distributions. Let us remark that in the first case, Λ is the feature function while, in the second case, the feature function is a random variable.

3.3 Decoding function Γ

Before turning to the definition of the decoding function Γ, let us focus first on the following simple question:

> *What is the best predictor of a real-valued random variable X, knowing only its distribution?*

The answer will obviously depend on the loss function considered and this will in turn influence the choice of the decoding function Γ, which purpose is to predict the random variable X. Assuming a L^r norm as loss function, the optimal predictor $\hat{\theta}_r(X)$ is then defined as

$$\hat{\theta}_r(X) \stackrel{\text{def}}{=} \operatorname*{argmin}_{c \in \mathbb{R}} \mathbb{E}_X[|X - c|^r].$$

In the case $r = 1$, the optimal predictor $\hat{\theta}_1(X)$ is simply the median of X; $r = 2$ corresponds to $\hat{\theta}_2(X) = \mathbb{E}[X]$, the mean value of X. In the following,

we call contextless prediction the X-prediction performed without other information than the distribution of X.

When focusing on the definition of the "inverse" mapping Γ, two natural definitions arise. When Λ is a bijection, the simplest predictor of X, given $b = \mathbb{P}(\sigma = 1)$, is $\Lambda^{-1}(b)$. Actually, it is the unique X-value such that $(\sigma \mid X = x)$ is distributed as $\mathbb{P}(\sigma = 1 \mid X = x) = b$, by definition (4) of Λ. We will denote this first choice for the decoding function

$$\Gamma^{\mathcal{L}} \stackrel{\text{def}}{=} \Lambda^{-1}.$$

$\Gamma^{\mathcal{L}}$ corresponds, in some sense, to a predictor based on maximum likelihood (ML). Indeed, suppose that the knowledge of $b = \mathbb{P}(\sigma = 1)$ is replaced with a sample of M independent copies s^k of a binary variable distributed as $\mathbb{P}(\sigma \mid X = x)$. The ML estimate of x is then $\Lambda^{-1}(\sum_k s^k/M)$. So the choice Λ^{-1} as decoding function corresponds to the ML estimate from a sample with an empirical rate of success equal to b.

In the general case of an increasing *corlol* encoding function, applying Jeffrey's rule (see Chan and Darwiche [6]), the cdf of X is updated using b to:

$$F^{\mathcal{J}}(x) = bF^1(x) + (1-b)F^0(x). \tag{8}$$

Let $X^{\mathcal{J}}$ be a random variable which distribution is $F^{\mathcal{J}}$. The predictor $\hat{\theta}(X^{\mathcal{J}})$ previously defined can be used irrespective of whether Λ is invertible or not. To refer to this second choice we will use the notation

$$\Gamma^{\mathcal{J}} \stackrel{\text{def}}{=} \hat{\theta}(X^{\mathcal{J}}).$$

Note that, while it may be costly in general to compute $\hat{\theta}(X^{\mathcal{J}})$, some choices of Λ lead to explicit formulas.

Mutual information. We consider here the case of the step function Λ^{MI} as encoding function. This function is of course not invertible and only the decoding function $\Gamma^{\mathcal{J}}$ can be used. Using (5) and (8), the cdf of $X^{\mathcal{J}}$ is

$$F^{\mathcal{J}}(x) = \begin{cases} 2(1-b)F(x), & \text{if } x \leq \mathrm{med}(X), \\ F^{\mathcal{J}}(\mathrm{med}(X)) + 2b\left(F(x) - F(\mathrm{med}(X))\right), & \text{if } x > \mathrm{med}(X). \end{cases} \tag{9}$$

In order to compute $\hat{\theta}_1(X^{\mathcal{J}})$, we need to solve the equation $F^{\mathcal{J}}(x) = 1/2$, leading to the decoding function

$$\Gamma^{\mathcal{J}}(b) = \begin{cases} F^{-1}\left(\frac{1}{4(1-b)}\right), & \text{if } b \leq \frac{1}{2}, \\ F^{-1}\left(\frac{4b-1}{4b}\right), & \text{if } b > \frac{1}{2}. \end{cases}$$

When F is not invertible, F^{-1} should be understood as the pseudo-inverse of F, commonly used to define quantiles:

$$F^{-1}(b) \stackrel{\text{def}}{=} \inf_x \{x \mid F(x) \geq b\}.$$

If we choose the predictor $\hat{\theta}_2$ based on a L^2 loss function, using the linearity of the expectation we get

$$\Gamma^{\mathcal{J}}(b) = \mathbb{E}\left[X^{\mathcal{J}}\right] = b\,\mathbb{E}\left[X \mid \sigma = 1\right] + (1-b)\mathbb{E}\left[X \mid \sigma = 0\right].$$

Max entropy principle. When one uses $\Gamma^{\mathcal{L}} = F^{-1}$ as decoding function, the contextless prediction, i.e. without any observation, is simply $F^{-1}\big(\mathbb{P}(\sigma = 1)\big)$. Moreover, we know that $\mathbb{P}(\sigma = 1) = \mathbb{E}[F(X)] = 1/2$—provided that X admits a pdf—so the ground prediction is the median of X. The choice $\Lambda = F$ and $\Gamma = F^{-1}$ is therefore optimal w.r.t. a L^1 loss function for the prediction error.

The other choice for the decoding function is to use $\Gamma^{\mathcal{J}}$ and to compute, for example, the predictor $\hat{\theta}_1(X^{\mathcal{J}})$. Using (5)–(8), we get the cdf of $X^{\mathcal{J}}$

$$F^{\mathcal{J}}(x) = \big((2b-1)F(x) - 2(b-1)\big)F(x),$$

and the sought function is solution of the following quadratic equation

$$\left((2b-1)F(x) - 2(b-1)\right)F(x) = \frac{1}{2},$$

with only one reachable root. Thus the decoding function $\Gamma^{\mathcal{J}}$ is

$$\Gamma^{\mathcal{J}}(b) = F^{-1}\left(\frac{2(b-1) + \sqrt{(2b-1)^2+1}}{4b-2}\right). \tag{10}$$

Let us remark that $\Gamma^{\mathcal{J}}$ is always more conservative for decoding than $\Gamma^{\mathcal{L}} = F^{-1}$, which can make predictions spanning the whole range $\mathcal{X}$ of possible outcomes. Figure 2 illustrates this.

Assume that two random variables X_1 and X_2 are equal with probability 1. Even if we build a latent model such that $\mathbb{P}(\sigma_1 = \sigma_2) = 1$, using $\Gamma^{\mathcal{J}}$ as decoding function will never predict $X_1 = X_2$. The approximation of the joint distribution of (X_1, X_2) with help of $\Gamma^{\mathcal{J}}$ is very rough when the variables are strongly dependent (see Proposition 4 in next section). However, the choice (F, F^{-1}) is equivalent to performing a X-quantiles regression. We will see in Section 7 that this last choice is better when variables are strongly dependent.

If one wishes to choose the decoding function based on the ML estimate $\Gamma^{\mathcal{L}} = \Lambda^{-1}$, it is of interest to generalize the max-entropy criterion in order to get an encoding function Λ with an optimal contextless prediction w.r.t.

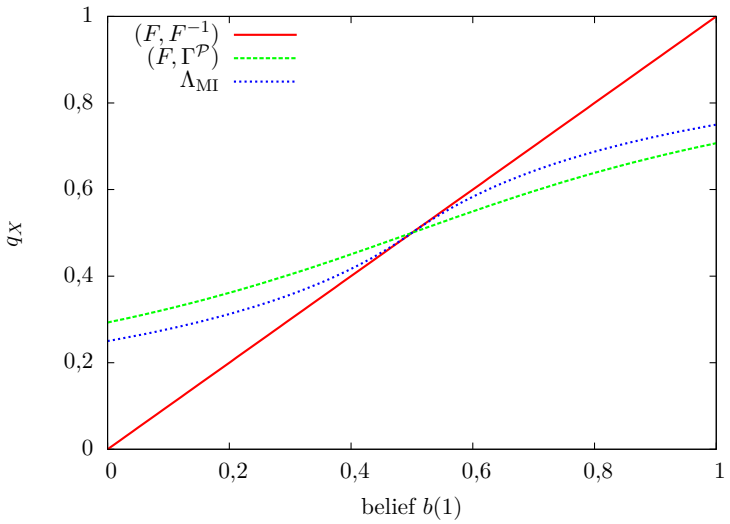


Figure 2: Prediction, expressed in quantiles, of the variable X_i for a given belief $b_i(1)$ on the associated latent binary variable σ_i depending on the choice of encoding/decoding functions. The left and right values are $(1/4, 3/4)$ for Λ^{MI} and $(1-\sqrt{2}/2, \sqrt{2}/2)$ for $(F, \Gamma^{\mathcal{J}})$.

a specific loss function. It is in fact quite simple to solve this problem and to obtain the sought encoding function which is based on the cdf (Martin [24, chapter 5]). We will use here only the cdf function because we are interested in L^1 error measure. Compared to a loss function based on the L^2 norm, it gives less weight to extreme values.

4 Building latent variables dependencies

It was shown in Section 3 how to relate the variable X_i to its latent state σ_i, by means of an encoding function Λ_i. The next issue concerns the dependencies at the latent state level and therefore the estimation of the underlying PGM on $\boldsymbol{\sigma}$. Assuming that we have only pairwise observations (1) and that we defined the σ_i's distributions conditionally to X_i's the natural way to go is to estimate pairwise marginals $\mathbb{P}(\sigma_i, \sigma_j)$. Using Jaynes' maximum entropy principle this leads us to the choice of a pairwise PGM for $\boldsymbol{\sigma}$ (see Appendix A). Given two real-valued variables X_i and X_j, with respective cdfs F_i and F_j, and two binary variables σ_i and σ_j, we will construct a pairwise model as described in Figure 1. The probability distribution of the vector $(X_i, X_j, \sigma_i, \sigma_j)$ for this model is

$$\mathbb{P}(X_i \leq x_i, X_j \leq x_j, \sigma_i = s_i, \sigma_j = s_j) = p_{ij}(s_i, s_j) F_i^{s_i}(x_i) F_j^{s_j}(x_j).$$

Since σ_i and σ_j are binary variables, $p_{ij}(s_i, s_j)$ can be expressed with 3 independent parameters:

$$\begin{aligned} p_i^1 &\stackrel{\text{def}}{=} \mathbb{P}(\sigma_i = 1) = \mathbb{E}(\sigma_i), \\ p_j^1 &\stackrel{\text{def}}{=} \mathbb{P}(\sigma_j = 1) = \mathbb{E}(\sigma_j), \\ p_{ij}^{11} &\stackrel{\text{def}}{=} \mathbb{P}(\sigma_i = 1, \sigma_j = 1) = \mathbb{E}(\sigma_i \sigma_j). \end{aligned}$$

The probability distribution is valid as soon as $(p_i^1, p_j^1) \in [0,1]^2$ and

$$p_{ij}^{11} \in \mathbb{D}(p_i^1, p_j^1) \stackrel{\text{def}}{=} \left[\max(0, p_i^1 + p_j^1 - 1), \min(p_i^1, p_j^1)\right].$$

Until now, we have been able to make optimal choices in some sense, but obviously the number of parameters is not enough to encode exactly any structure of dependency. This is shown in the following proposition.

Proposition 4. *When the mutual information $I_{\mathcal{P}}(X_i, X_j)$ between the real variables is strictly greater than* $\log(2)$*, our model is not able to perfectly encode the joint distribution of X_i and X_j for any choice of encoding function.*

This result is compatible with intuition: whatever the definition of the binary variables, it will not be possible to share more than one bit of information between two of them. However, we shall see in Section 7 that it is still possible to obtain quasi-optimal performances for the prediction task even when the mutual information is strictly greater than log(2).

Proof. See Appendix B. ■

We will first propose a simple EM-based algorithm to estimate the pairwise distribution p_{ij} of (σ_i, σ_j), without discussing how to estimate the joint distribution of $\boldsymbol{\sigma}$ from them. We will come back to this problem in the end of this section.

4.1 Pairwise distributions estimation

The choice of the encoding functions Λ_i imposes the marginal distributions of the latent variables σ_i; indeed we have seen that

$$p_i^1 = \mathbb{P}(\sigma_i = 1) = \mathbb{E}[\Lambda_i(X_i)] = \int_{\mathcal{X}_i} \Lambda_i(x) dF_i(x). \tag{11}$$

These parameters can easily be estimated using empirical moments and it will only remain to estimate the correlation parameter p_{ij}^{11}. We propose here to carry out a maximum likelihood estimation. The estimation of each parameter p_{ij}^{11} is independent of the others and we carry out one unidimensional likelihood maximization per edge. For the sake of simplicity, we assume that the random variables admit probability distribution functions.

The joint pdf of (X_i, X_j) associated with the distribution p_{ij} will be referred to as

$$f^{ij}_{p_{ij}}(x_i, x_j) \stackrel{\text{def}}{=} \sum_{s_i, s_j} p_{ij}(s_i, s_j) f_i^{s_i}(x_i) f_j^{s_j}(x_j),$$

where $f_i^{s_i}$ is the pdf associated to $dF_i^{s_i}$. Let us first express the logarithm of the likelihood of a distribution p_{ij} of (σ_i, σ_j) corresponding to the pairwise observations $\mathbf{x}$ described in (1).

$$\begin{aligned} L(\mathbf{x}, p_{ij}) &= \sum_{k=1}^{N_{ij}} \log f^{ij}_{p_{ij}}(x_i^k, x_j^k) \\ &= \sum_{k=1}^{N_{ij}} \log \left(\sum_{s_i, s_j} p_{ij}(s_i, s_j) f_i^{s_i}(x_i^k) f_j^{s_j}(x_j^k) \right). \end{aligned}$$

Because of the hidden variables σ_i and σ_j, a sum appears within the logarithms. Therefore, it will not be possible to find explicitly the distributions p_{ij} maximizing $L(\mathbf{x}, p_{ij})$. The usual approach is to use the EM algorithm of Dempster et al. [10]. It consists in building a sequence of (σ_i, σ_j)-distribution $p_{ij}^{(n)}$ with increasing likelihood. Using the following notation

$$p_{ij}^{(n)}(s_i, s_j \mid x_i, x_j) \stackrel{\text{def}}{=} \mathbb{P}^{(n)}(\sigma_i = s_i, \sigma_j = s_j \mid X_i = x_i, X_j = x_j),$$

the EM algorithm can be expressed as

$$p_{ij}^{(n+1)} \leftarrow \underset{p_{ij}}{\operatorname{argmax}} \quad \ell(p_{ij} \| p_{ij}^{(n)}) \stackrel{\text{def}}{=} \sum_k \sum_{s_i, s_j} p_{ij}^{(n)}(s_i, s_j \mid x_i^k, x_j^k) \log p_{ij}(s_i, s_j),$$

The derivative of $\ell(p_{ij} \| p_{ij}^{(n)})$ with respect to p_{ij}^{11} is

$$\frac{\partial \ell(p_{ij} \| p_{ij}^{(n)})}{\partial p_{ij}^{11}} = \sum_{k=1}^{N_{ij}} \sum_{s_i, s_j} \left(2 \mathbb{1}_{\{s_i = s_j\}} - 1 \right) \frac{p_{ij}^{(n)}(s_i, s_j \mid x_i^k, x_j^k)}{p_{ij}(s_i, s_j)},$$

Stationary points yield an obvious solution, which is

$$p_{ij}(s_i, s_j) = \frac{1}{N_{ij}} \sum_{k=1}^{N_{ij}} p_{ij}^{(n)}(s_i, s_j \mid x_i^k, x_j^k).$$

The function that we maximize being concave, this solution is the unique stationary point of $\ell(p_{ij} \| p_{ij}^{(n)})$. We obtain the following update rule for the EM algorithm

$$p_{ij}^{(n+1)}(1,1) \leftarrow \frac{1}{N_{ij}} \sum_{k=1}^{N_{ij}} \frac{\psi_{ij}^{(n)}(1,1) \Lambda_i(x_i^k) \Lambda_j(x_j^k)}{Z_{ij}(x_i^k, x_j^k)}, \tag{12}$$

with

$$\psi_{ij}^{(n)}(s_i, s_j) \overset{\text{def}}{=} \frac{p_{ij}^{(n)}(s_i, s_j)}{p_i^{(n)}(s_i) p_j^{(n)}(s_j)},$$

$$Z_{ij}(x_i, x_j) \overset{\text{def}}{=} \sum_{s_i, s_j} \psi_{ij}^{(n)}(s_i, s_j) \Lambda_i^{s_i}(x_i) \Lambda_j^{s_j}(x_j),$$

and $\Lambda^1 \overset{\text{def}}{=} \Lambda$, $\Lambda^0 \overset{\text{def}}{=} 1 - \Lambda$. The update rule (12) is quite simple, although one has to check that the estimated parameter is valid, i.e. $p_{ij}^{11} \in \mathbb{D}(p_i^1, p_j^1)$. If it is not the case, it means that the parameter saturates at one bound.

4.2 Latent binary PGM estimation compatible with BP

Now that the pairwise marginals of the model are estimated, it remains to use them to determine the joint distribution $p_{\boldsymbol{\sigma}}$ of $\boldsymbol{\sigma}$. First, let us remark that (as discussed by Mackay et al. [23]) having compatible marginals does not guarantee the existence of a joint distribution $p_{\boldsymbol{\sigma}}$ such as

$$\forall (i,j) \in \mathbb{E}, \forall s_i, \sum_{\mathbf{s}_{\mathbb{V}\setminus\{i,j\}}} p_{\boldsymbol{\sigma}}(\boldsymbol{\sigma} = \mathbf{s}) = p_{ij}(s_i, s_j),$$

However, in the case where the graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ contains no cycles, the joint distribution is entirely determined by its pairwise marginals. This joint distribution is expressed as

$$p_{\boldsymbol{\sigma}}(\boldsymbol{\sigma} = \mathbf{s}) = \prod_{(i,j)\in\mathbb{E}} \frac{p_{ij}(s_i, s_j)}{p_i(s_i) p_j(s_j)} \prod_{i\in\mathbb{V}} p_i(s_i), \tag{13}$$

with $p_i(s_i) = \sum_{s_j} p_{ij}(s_i, s_j)$.

In the general case of a graph containing cycles, the situation is more complex. This inverse problem is much studied in statistical physics (see Cocco and Monasson [7] and references within). Potentially it is NP-hard and may have no solution. Only approximate methods can be used for graph of large size. Wainwright [37] proposed an approach of particular interest, which takes into account the fact that once the distribution $p_{\boldsymbol{\sigma}}$ is fixed in an approximate way, the marginalization will also be performed in an approximate way. The idea is to use compatible approximations for these two tasks. In our case, we wish to use the BP algorithm, described in forthcoming Section 5, to compute the approximate marginals of $p_{\boldsymbol{\sigma}}$. It seems reasonable to impose that, without any observation, the answer given by BP is the historical marginals $\{p_{ij}\}$ and $\{p_i\}$. For doing so, the distribution $p_{\boldsymbol{\sigma}}$ should be chosen under the Bethe approximation (13) which is closely related to the BP algorithm, as we shall see in Section 5. If this choice is a good candidate as starting point, the Bethe approximation is usually too rough and overestimates correlations, and it is thus necessary to improve on it. This can be achieved using various results from linear

response theory (see Welling and Teh [38], Yasuda and Tanaka [39], Mézard and Mora [26]), when the level of correlation is not too high.

We use instead a simple but more robust approach, which is to modify the model using a single parameter α such as

$$p_{\boldsymbol{\sigma}}(\boldsymbol{\sigma} = \mathbf{s}) = \prod_{(i,j)\in\mathbb{E}} \left(\frac{p_{ij}(s_i, s_j)}{p_i(s_i)p_j(s_j)} \right)^{\alpha} \prod_{i\in\mathbb{V}} p_i(s_i). \tag{14}$$

α can roughly be interpreted as an inverse temperature, which role is to avoid overcounting interactions when the graph contains cycles. This parameter can easily be calibrated by finding a phase transition w.r.t. α. Indeed, for $\alpha = 0$, the BP output is exactly $\{p_i\}$ and, when α increases, it remains close to it until some discontinuity appears (see Furtlehner et al. [15]). In some sense, the best α corresponds to the maximal interaction strength such that the BP output remains close to $\{p_i\}$.

5 A message passing inference algorithm

According to the results of Section 3, observations about the real-valued random variable X_i are converted into knowledge of the marginal distribution of σ_i. In order to estimate the distributions of the other binary latent variables, we need an inference algorithm that imposes this marginal constraint to node i when X_i is observed. For this task, we propose a modified version of the BP algorithm. This extends and simplifies a BP variant proposed by Teh and Welling [36]. Moreover, we provide sufficient conditions for convergence of our algorithm (Proposition 5).

5.1 The BP algorithm

We present here the BP algorithm, first described by Pearl [30], in a way very similar to the one of Yedidia et al. [40]. We use in this section a slightly more general notation than in Section 1, since instead of considering only pairwise interactions, variables in the set $\mathbb{V}$ interact through factors, which are subsets $a \subset \mathbb{V}$ of variables. If $\mathbb{F}$ is this set of factors, we consider the following probability measure

$$\mathbb{P}(\boldsymbol{\sigma} = \mathbf{s}) = \prod_{a\in\mathbb{F}} \psi_a(\mathbf{s}_a) \prod_{i\in\mathbb{V}} \phi_i(s_i), \tag{15}$$

where $\mathbf{s}_a = \{s_i, i \in a\}$. It is also possible to see variables and factors as nodes of a same bipartite graph, in which case the shorthand notation $i \in a$ should be interpreted as "there is an edge between i and a". $\mathbb{F}$ together with $\mathbb{V}$ define a factor graph, such as defined by Kschischang et al. [22]. The set $\mathbb{E}$ of edges contains all the couples $(a, i) \in \mathbb{F} \times \mathbb{V}$ such that $i \in a$. We denote by d_i the degree of the variable node i. The BP algorithm is a message

passing procedure, whose output is a set of estimated marginal probabilities, the beliefs $b_a(\mathbf{s}_a)$ (including single nodes beliefs $b_i(s_i)$). The idea is to factor the marginal probability at a given site as a product of contributions coming from neighboring factor nodes, which are the messages. With definition (15) of the joint probability measure, the updates rules read:

$$m_{a\to i}(s_i) \leftarrow \sum_{\mathbf{s}_{a\setminus i}} \psi_a(\mathbf{s}_a) \prod_{j\in a\setminus i} n_{j\to a}(s_j), \tag{16}$$

$$n_{i\to a}(s_i) \stackrel{\text{def}}{=} \phi_i(s_i) \prod_{a'\ni i, a'\neq a} m_{a'\to i}(s_i), \tag{17}$$

where the notation $\sum_{\mathbf{s}_a}$ should be understood as summing all the variables σ_i, $i \in a \subset \mathbb{V}$, over the realizations $s_i \in \{0, 1\}$. In practice, the messages are often normalized so that $\sum_{s_i} m_{a\to i}(s_i) = 1$.

At any point of the algorithm, one can compute the current beliefs as

$$b_i(s_i) \stackrel{\text{def}}{=} \frac{1}{Z_i} \phi_i(s_i) \prod_{a\ni i} m_{a\to i}(s_i), \tag{18}$$

$$b_a(\mathbf{s}_a) \stackrel{\text{def}}{=} \frac{1}{Z_a} \psi_a(\mathbf{s}_a) \prod_{i\in a} n_{i\to a}(s_i), \tag{19}$$

where Z_i and Z_a are normalization constants that ensure that

$$\sum_{\sigma_i} b_i(\sigma_i) = 1, \qquad \sum_{\boldsymbol{\sigma}_a} b_a(\boldsymbol{\sigma}_a) = 1. \tag{20}$$

When the algorithm has converged, the obtained beliefs b_a and b_i are compatible:

$$\sum_{\mathrm{s}_{a\setminus i}} b_a(\mathrm{s}_a) = b_i(s_i). \tag{21}$$

Yedidia et al. [40] proved that the belief propagation algorithm is an iterative way of solving a variational problem: namely it minimizes the Kullback-Leibler divergence $D_{\mathrm{KL}}(b\|p)$ to the true probability measure (15) over all Bethe approximations on the factor graph, of the form

$$b(\mathbf{s}) = \prod_{a\in\mathbb{F}} \frac{b_a(\mathbf{s}_a)}{\prod_{i\in a} b_i(s_i)} \prod_{i\in\mathbb{V}} b_i(s_i),$$

subject to constraints (20)–(21). The approximation is actually exact when the underlying graph is a tree. The stationary points of the above variational problem are beliefs at a fixed point of the BP algorithm (see Yedidia et al. [40]). This variational description of BP will be used in the next section to derive a new variant of the algorithm.

5.2 Imposing beliefs: mirror BP

In the following, $\mathbb{V}^*$ will be the set of nodes i such that X_i is observed:

$$X_i = x_i^*, \text{ for all } i \in \mathbb{V}^*. \tag{22}$$

Assuming that the model (ψ_a and ϕ_i) is given, we wish to include in the algorithm some constraints on the beliefs of the form

$$\forall i \in \mathbb{V}^*, \forall s_i \in \{0,1\}, b_i(s_i) = b_i^*(s_i). \tag{23}$$

We suppose in the following that each b_i^* is normalized. The issue of how to convert real-valued observation to this distribution b_i^* has been studied in Section 3. We seek to obtain a new update rule from the Kullback-Leibler divergence minimization, with the additional constraints (23). Constraints of this form as sometimes referred to as soft constraints in the Bayesian community [4].

We start from the Lagrangian of the minimization problem:

$$\begin{aligned}\mathcal{L}(b,\lambda) = D_{\mathrm{KL}}(b\|p) &+ \sum_{\substack{i\in\mathbb{V}\setminus\mathbb{V}^* \\ a\ni i, s_i}} \lambda_{ai}(s_i)\Big(b_i(s_i) - \sum_{\mathbf{s}_{a\setminus i}} b_a(\mathbf{s}_a)\Big) \\ &+ \sum_{\substack{i\in\mathbb{V}^* \\ a\ni i, s_i}} \lambda_{ai}(s_i)\Big(b_i^*(s_i) - \sum_{\mathbf{s}_{a\setminus i}} b_a(\mathbf{s}_a)\Big) + \sum_{i\in\mathbb{V}} \gamma_i \left(\sum_{s_i} b_i(s_i) - 1\right),\end{aligned}$$

with $D_{\mathrm{KL}}(b\|p)$ defined as

$$D_{\mathrm{KL}}(b\|p) \stackrel{\text{def}}{=} \sum_{a,\boldsymbol{s}_a} b_a(\boldsymbol{s}_a) \log \frac{b_a(\boldsymbol{s}_a)}{\psi_a(\boldsymbol{s}_a)} + \sum_{i,s_i} b_i(s_i) \log \frac{b_i(s_i)^{1-d_i}}{\phi_i(s_i)}.$$

The stationary points satisfy

$$\begin{cases} b_a(\mathbf{s}_a) = \psi_a(\mathbf{s}_a) \exp\Big(\sum_{i\in a} \lambda_{ai}(s_i) - 1\Big), \forall a \in \mathbb{F}, \\ b_i(s_i) = \phi_i(s_i) \exp\Big(\frac{\sum_{a\ni i} \lambda_{ai}(s_i)}{d_i - 1} + 1 - \gamma_i\Big), \forall i \notin \mathbb{V}^*, \\ b_i(s_i) = b_i^*(s_i), \forall i \in \mathbb{V}^*. \end{cases}$$

Following Yedidia et al. [40], we introduce the parametrization

$$\lambda_{ai}(s_i) = \log n_{i\to a}(s_i),$$

for all edges $(ai) \in \mathbb{E}$. Note that we do not consider any node in $\mathbb{V}\setminus\mathbb{V}^*$ of degree d_i equal to 1 since they play no role in the minimization problem. For all nodes $i \notin \mathbb{V}^*$ we also have

$$\lambda_{ai}(s_i) = \log\Big[\phi_i(s_i) \prod_{b\ni i, b\neq a} m_{b\to i}(s_i)\Big].$$

Then it follows that, whenever $i \notin \mathbb{V}^*$,

$$n_{i\to a}(s_i) = \phi_i(s_i) \prod_{b\ni i, b\neq a} m_{b\to i}(s_i).$$

Enforcing the compatibility constraints on nodes $i \notin \mathbb{V}^*$ shows that update rules (16)–(17) are still valid for these nodes. For $i \in \mathbb{V}^*$, the compatibility constraints yield

$$\begin{aligned} b_i^*(s_i) &= \sum_{\mathbf{s}_{a\setminus i}} b_a(\mathbf{s}_a) = \sum_{\mathbf{s}_{a\setminus i}} \psi_a(\mathbf{s}_a) \prod_{j\in a} n_{j\to a}(s_j) \\ &= n_{i\to a}(s_i) \sum_{\mathbf{s}_{a\setminus i}} \psi_a(\mathbf{s}_a) \prod_{j\in a} n_{j\to a}(s_j). \end{aligned}$$

Until now the message from a factor a to a variable $i \in \mathbb{V}^*$ has not been defined. For convenience we define it as in (16) and the preceding equation becomes

$$n_{i\to a}(s_i) m_{a\to i}(s_i) = b_i^*(s_i),$$

as in the usual BP algorithm. This leads to a definition that replaces (17) when $i \in \mathbb{V}^*$

$$n_{i\to a}(s_i) = \frac{b_i^*(s_i)}{m_{a\to i}(s_i)} = \frac{b_i^*(s_i)}{b_i(x_i)} \phi_i(s_i) \prod_{b\ni i, b\neq a} m_{b\to i}(s_i). \tag{24}$$

Therefore the message (24) is the BP message (17) multiplied by the ratio of the belief we are imposing over the current belief computed using (18). This is very similar to iterative proportional fitting (IPF, see Darroch and Ratcliff [9]). To sum up, the characteristics of this new variant of belief propagation are:

- all factors and all variables which values have not been fixed send the same messages (16)–(17) as in classic BP;
- variables which value has been fixed use the new messages (24);
- beliefs for factors or for variables which value has not been fixed are still computed using (18)–(19);

In the classical BP algorithm, the information sent by one node can only go back to itself through a cycle of the graph. When (24) is used, however, the variable with fixed value acts like a mirror and sends back the message to the factor instead of propagating it through the graph. It is to emphasize this property that we call our new method the *mirror BP* (mBP) algorithm. Note that it could be defined for variables valued in any discrete alphabet.

A very similar algorithm to our mBP has been proposed by Teh and Welling [36]. Their algorithm is described as iterations of successive BP

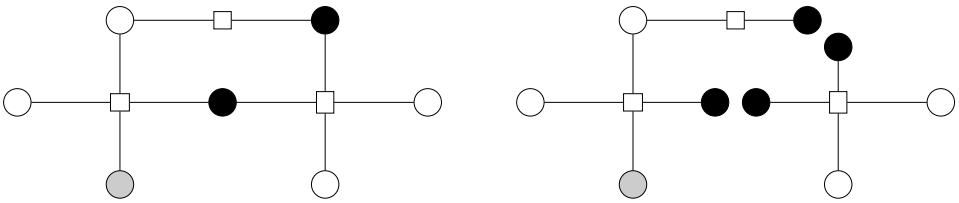


Figure 3: Illustration of Proposition 5. If only black nodes are in $\mathbb{V}^*$, Proposition 5 tells us that mBP converges since the resulting graph $\mathcal{T}(\mathcal{G}, \mathbb{V}^*)$ (right graph) contains two disconnected trees with exactly two nodes in $\mathbb{V}^*$. If we add the gray node in $\mathbb{V}^*$ then Proposition 5 does not apply, the right tree contains three nodes in $\mathbb{V}^*$, and we cannot conclude about convergence. However, on the other part of the graph Proposition 5 still holds.

runs on unobserved nodes and IPF on nodes in $\mathbb{V}^*$. The update (24) is just obtained as direct IPF. The main drawback of their version is that it assumes a particular update ordering because they consider that the updates (17)–(24) are of different nature, which is in fact not really necessary and complicates its use.

It is known that BP can exhibit non convergent behavior in loopy networks, although sufficient conditions for convergence are known (see e.g. Mooij and Kappen [29], Tatikonda and Jordan [35], Ihler et al. [18]). Since the mirroring behavior of our algorithm seems to be quite different, we present some sufficient conditions for convergence.

Definition 1. *Let $\mathcal{T}(\mathcal{G}, \mathbb{V}^*)$ be the factor graph where each node $i \in \mathbb{V}^*$ has been cloned d_i times, each clone being attached to one (and only one) neighbor of i. We denote $\mathcal{T}(\cdot, \mathbb{V}^*) : \mathcal{G} \to \mathcal{T}(\mathcal{G}, \mathbb{V}^*)$, the transformation applied to a factor graph $\mathcal{G}$ for a given set of variable nodes $\mathbb{V}^*$.*

Example of a such transformation $\mathcal{G} \to \mathcal{T}(\mathcal{G}, \mathbb{V}^*)$ is shown in Figure 3. The following proposition describes cases where the mBP algorithm is guaranteed to converge.

Proposition 5. *If the graph $\mathcal{T}(\mathcal{G}, \mathbb{V}^*)$ is formed by disconnected trees containing not more than two leaves cloned from $\mathbb{V}^*$, the mBP algorithm is stable and converges to a unique fixed point.*

Proof. See Appendix. ■

5.3 Pairwise case

The mBP algorithm is actually simpler for a pairwise Markov random field like (3). Indeed the factors are simply the edges $(i, j) \in \mathbb{E}$ and we can use

the following notation for the messages:

$$m_{j\to i}(s_i) \stackrel{\text{def}}{=} m_{(i,j)\to i}(s_i), \qquad n_{i\to j}(s_i) \stackrel{\text{def}}{=} n_{i\to(i,j)}(s_i).$$

The message update rules (16)–(17) reduce to

$$m_{j\to i}(s_i) \leftarrow \sum_{s_j} \psi_{ij}(s_i, s_j) n_{j\to i}(s_j), \tag{25}$$

$$n_{i\to j}(s_i) = \begin{cases} \phi_i(s_i) \displaystyle\prod_{(k,i)\in\mathbb{E}, k\neq j} m_{k\to i}(s_i), & \forall i \in \mathbb{V}\setminus\mathbb{V}^* \\ \dfrac{b_i^*(s_i)}{m_{j\to i}(s_i)}, & \forall i \in \mathbb{V}^*. \end{cases} \tag{26}$$

The marginal and joint beliefs formula (18)–(19) respectively read

$$b_i(s_i) = \frac{1}{Z_i}\phi_i(s_i) \prod_{(j,i)\in\mathbb{E}} m_{j\to i}(s_i), \quad \forall i \in \mathbb{V}\setminus\mathbb{V}^*, \tag{27}$$

$$b_{ij}(s_i, s_j) = \frac{1}{Z_{ij}}\psi_{ij}(s_i, s_j) n_{i\to j}(s_i) n_{j\to i}(s_j), \quad \forall (i,j) \in \mathbb{E}.$$

This is the form of mBP that is used in the remainder of this paper.

6 Implementation aspects

At this point, it is useful to gather the various elements discussed previously in view of practical implementation. The workflows for the main elements of our framework are discussed below.

Encoding/decoding Selecting Λ and Γ is the first step. There are three possibilities:

- encoding via Λ^{MI} (7) and decoding with $\Gamma^{\mathcal{J}}$ (9);
- encoding via cdf F and decoding with F^{-1};
- encoding via cdf F and decoding with $\Gamma^{\mathcal{J}}$ (10).

Model set-up. It is done in several steps, using as input historical data of the form (1).

1. for each variable $i \in \mathbb{V}$, estimate Λ_i (either the empirical cdf or the empirical median).
2. for each variable $i \in \mathbb{V}$, compute p_i^1 according to (11); in practice, for the choices of Λ above, one can set $p_i^1 = 0.5$.
3. for each pair of variables $(i,j) \in \mathbb{E}$, compute iteratively p_{ij}^{11} using (12).
4. the joint binary model is then given by (14).

Online inference. This is an endless loop, which basic step goes as follows:

1. get new data of the form (22) and update $\mathbb{V}^*$; depending on the problem at hand, it can be chosen to discard earlier data according to some obsolescence criterion;
2. compute the imposed beliefs (23) using the encoding function Λ.
3. run mBP on $\mathcal{G}$ using (25) and (26) until convergence;
4. compute the beliefs $b_i, i \in \mathbb{V} \setminus \mathbb{V}^*$ using (27);
5. compute the prediction of $X_i, i \in \mathbb{V} \setminus \mathbb{V}^*$ using Γ_i.

Before moving to numerical experiments, let us describe the algorithmic complexity of our method. The relevant parameters and constants are

- $|\mathbb{V}|$ the number of variables,
- $|\mathbb{E}|$ the number of edges,
- η^{a} the maximum number of iterations for the algorithm a,
- N the total number of pairwise observations,
- N_i the number of observations of the variable X_i,
- N_{ij} the number of joint observations of the pair (X_i, X_j),
- $\langle \cdot \rangle$ the mean value over all the possibilities.

Table 1 summarizes the global complexity of all the steps described in previous sections. The empirical cdf of the variable X_i is stored as a vector $\mathbf{h}^i$ of length ℓ_h such that $F(h_k^i) = \frac{k}{\ell_h}$ and is built by sorting the vector of observations. Given an observation $X_i = x_i$, the encoded belief can thus be computed through a dichotomy search on $\mathbf{h}^i$. Decoding beliefs is immediate.

Let us first remark that all the steps can be easily parallelized, since computations over nodes or edges can be made independently. We can clearly see that the crucial steps are the mirror-BP algorithm and the pairwise marginal computations, which have both a complexity linear in the number of edges. While the constant η^{EM} is usually smaller than 10 and can be neglected, η^{BP} has to be controlled more carefully. The overall complexity of model estimation scales with the number of edges, which is unavoidable, and is linear in the number of pairwise observations per edge, which seems reasonable. For the inference part, neither encoding or decoding add an additional complexity cost and we just have to be careful to obtain a model which converges quickly. This is not a problem as long as the model is not both dense and highly correlated.

Task	Step	Global complexity
Model estimation	Encoding functions	$\mathcal{O}(\lvert\mathbb{V}\rvert \langle N_i \log N_i \rangle)$
	Pairwise marginals $\hat{p}_{ij}$	$\mathcal{O}(\lvert\mathbb{E}\rvert \eta^{\mathrm{EM}} \langle N_{ij} \rangle) = \mathcal{O}(\eta^{\mathrm{EM}} N)$
Inference	Encoding	$\mathcal{O}(\lvert\mathbb{V}\rvert \log \ell_h)$
	mBP based inference	$\mathcal{O}(\lvert\mathbb{E}\rvert \eta^{\mathrm{BP}})$
	Decoding	$\mathcal{O}(\lvert\mathbb{V}\rvert)$

Table 1: Complexity of each step of the method described in the previous sections.

7 Numerical experiments

In order to understand its behavior, we test our method on two classes of models with increasing complexity

- generic models: pairs (X_1, X_2) of real-valued random variables, trees with interior connectivity fixed;
- road traffic networks: first a synthetic model that demonstrates the scalability of the proposed method, and then one built from actual traffic data about the Paris urban area.

For each case, we repeat the following *decimation experiment*: for an outcome of the random vector $\mathbf{X}$, we observe its components X_i in a random order and we make prediction about unobserved components using our method. This stacks up the performance of the different choices of encoding and decoding functions against each other when the proportion of observed variables varies.

As discussed in Section 6, the choices for the encoding and decoding functions are:

- the step function Λ^{MI} with the decoding function $\Gamma^{\mathcal{J}}$ of (9),
- the cumulative distribution function F with its inverse $\Gamma^{\mathcal{L}} = F^{-1}$,
- the cumulative distribution function F with the decoding function $\Gamma^{\mathcal{J}}$ of (10).

Each of these choices yields an estimator θ for which we will compute the performance w.r.t. the L^1 norm:

$$\mathbb{E}_X\Big[|\theta(X) - X|\Big]. \tag{28}$$

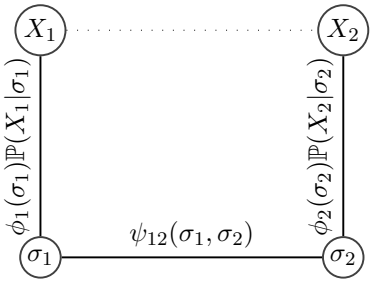


Figure 4: Model of the random vector $(X_1, X_2, \sigma_1, \sigma_2)$. The true distribution of (X_1, X_2) is approximated through the latent variables σ_1 and σ_2.

7.1 Generic models

These synthetic models are based on Gaussian copulas with support corresponding to one of the three cases previously described. More precisely, it corresponds to the support of the precision matrix, i.e. the inverse covariance matrix, of the Gaussian vector $\mathbf{Y}$. For doing so, we randomly generate the partial correlations, the entries of the precision matrix of $\mathbf{Y}$, with uniform random variables on $[-1, -0.2] \cup [0.2, 1]$. Since this will not always lead to a positive definite precision matrix, we use this matrix as a starting point and reduce the highest correlation until it becomes definite positive.

We can then generate outcomes of this Gaussian vector $\mathbf{Y}$ and transform them, using the function which maps a Gaussian variable $\mathcal{N}(0, 1)$ into a variable of chosen cdf F_X. More precisely, each component of the vector $\mathbf{X}$ is defined as

$$X_i = F_X^{-1}\left(F_{\mathcal{N}(0,1)}(Y_i)\right),$$

where $F_{\mathcal{N}(0,1)}$ is the cdf of a $\mathcal{N}(0, 1)$ variable. Exact inference can then be performed using the Gaussian vector $\mathbf{Y}$ since the $\mathbf{X}$-dependencies are based on a Gaussian copula.

We will sometimes consider the case of $\beta(a, b)$ random variables, so let us recall their pdfs $f^{\beta}_{a,b}$ for $a, b \in]0, +\infty[$

$$f^{\beta}_{a,b}(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}\mathbb{1}_{[0,1]}(x),$$

where the normalization constant $B(a, b)$ is the Euler beta function. These distributions are of particular interest because different cases arise depending on the parameters a and b. Indeed, it is possible to obtain almost binary $(a, b \to 0)$, unimodal $(a, b > 1)$ or uniform $(a, b = 1)$ distributions on $[0, 1]$.

A pair (X_1, X_2) of real-valued random variables. Let us start with the simple case where the vector $\mathbf{X}$ is just two random real-valued variables (Figure 4). We repeat 100,000 times the decimation experiment. In this

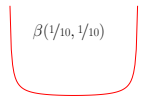
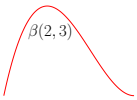
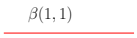
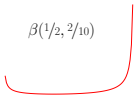
Density	Correlation	$\hat{\theta}_1$	(F, F^{-1})	$(F, \Gamma^{\mathcal{J}})$	Λ^{MI}
	$\rho = 0.5$	0.320 0	**0.2%** −0.0034	3.1% −0.0079	3.3% 0.0001
	$\rho = 0.9$	0.136 0	**0.1%** −0.0015	22% −0.0072	9.5% −0.0006
	$\rho = 0.5$	0.142 0	**1.4%** −0.0026	4.4% 0.0047	6.7% 0.0044
	$\rho = 0.9$	0.069 0	**1.3%** 0.001	68.8% 0.0125	61.7% 0.0092
	$\rho = 0.5$	0.210 0	**0.1%** 0.0002	4.6% 0.0001	7.3% −0.0003
	$\rho = 0.9$	0.098 0	**0.4%** 0.0002	66.2% −0.0009	58.6% 0
	$\rho = 0.5$	0.230 0	**2.7%** 0.069	4.4% 0.051	7.5% −0.041
	$\rho = -0.7$	0.182 0	**4.1%** 0.057	16.9% −0.088	20.8% −0.052

Table 2: Performances of all estimators in the case of Figure 4. The first line is mean L^1 error in % of deviation from the optimal performance obtained with $\hat{\theta}_1(X)$, the conditional median predictor, displayed in the third column. The second one is the estimator biases. Bold values are the best performing choices.

case, this experiment is just to observe either X_1 or X_2 for a given outcome of the vector (X_1, X_2) and to predict the other variable. In addition to the L^1 performance, we compute the biases of the different estimators θ

$$\mathbb{E}_X \left[\theta(X) - \hat{\theta}_1(X) \right]. \tag{29}$$

We recall that $\hat{\theta}_1(X)$ is the optimal predictor w.r.t. the L^1 distance i.e. the conditional median.

The results, for various values of a, b and $\rho \stackrel{\text{def}}{=} \text{cov}(Y_i, Y_j)$, are given in Table 2. The first line is the L^1 performance (28) and the second one the bias (29) of the estimator. Generally, with weak correlations all estimators have a satisfactory behavior. However the best choice is the cdf function $\Lambda = F$ with the inverse mapping $\Gamma^{\mathcal{L}} = F^{-1}$. As expected, the conservative property of the decoding $\Gamma^{\mathcal{J}}$ is a real drawback in the case of strong correlation because it prevents from predicting extreme values (see Figure 2). When considering symmetric variables X_i, all estimators biases are close to 0. Even with $\beta(2, 3)$ variables, this bias is negligible. In the case of asymmetrical variables, these biases are clearly non zero, but do not prevent from obtaining good performance.

Regular trees. We consider here the case of a tree containing 100 nodes with given connectivity n for interior nodes: all nodes, except leaf nodes, have exactly n neighbors. For $n = 3$, we get a binary tree with maximal depth equal to 6. We perform the decimation experiments and results are presented in Figure 5. For the sake of comparison, we show three other predictors: the median (in red), the k nearest neighbors (k-NN, Cover and Hart [8]) predictor (in orange) and the perfect predictor (in black), which is obtained by computing the conditional mean of the vector $\mathbf{Y}$. The k-NN predictor is manually optimized to the $k = 50$ nearest neighbors in the whole training set used to build the model. This predictor is known to perform reasonably on road traffic data (Smith et al. [33]), but its complexity is too high for large networks compared to BP. Moreover, it requires complete observations of the network, which are not available when dealing with probe vehicles data. However, on synthetic models, we can use it as a comparison.

As a general rule, the choice (F, F^{-1}) seems to be the best one. Let us remark that, if we continue to increase the connectivity n, this situation can change. In fact, at very high connectivity ($n \sim 10$), the convergence of mBP can be very slow. Non convergent cases can then impact the result and one should rather use Λ^{MI}. Indeed, for the choice Λ^{MI} the mBP algorithm is strictly equivalent to BP. In this case, the BP algorithm is more stable since it is always converging on trees. At this point, we discard the choice $(F, \Gamma^{\mathcal{J}})$ which is clearly inferior to the other ones.

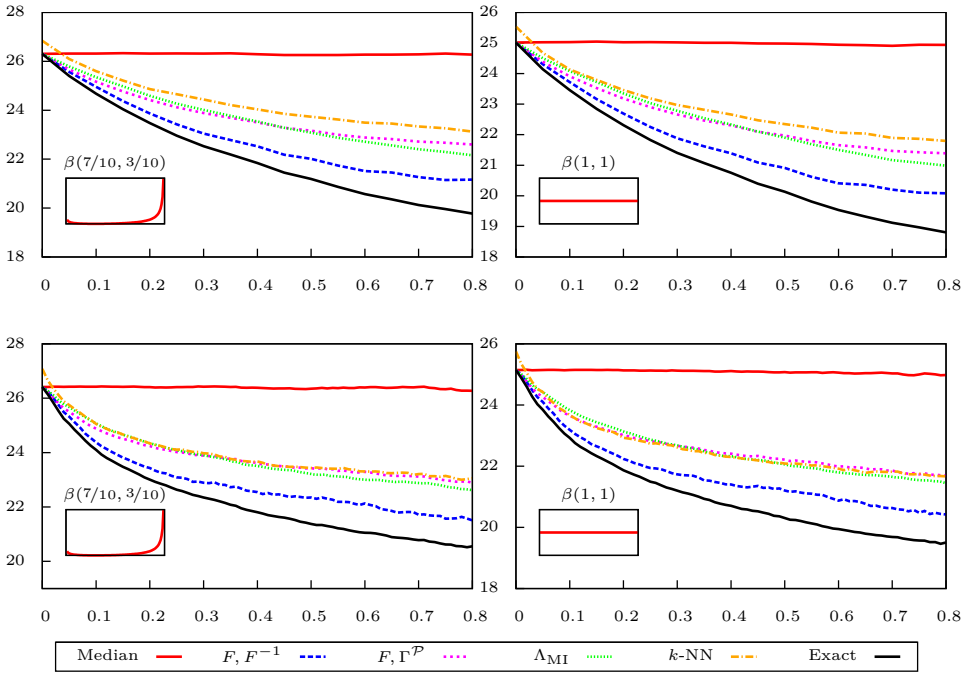


Figure 5: Mean L^1 prediction error of unobserved variables ($\times 100$) as a function of the proportion of revealed variables; the small embedded figures are the corresponding pdf of the beta variables. The connectivity is $n = 3$ for top figures and $n = 5$ for the bottom ones. Each tree contains 100 variables.

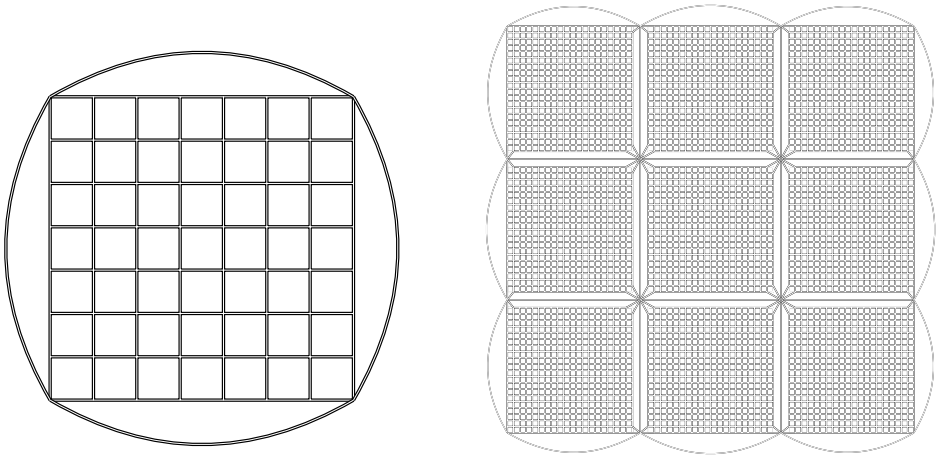


Figure 6: Simple models of urban road networks with two-way streets. The inner grids represent the city center, while exterior edges form ring roads connecting them. Left network contains 232 nodes and 624 edges while the the one on the right contains 13,752 nodes and 39,880 edges.

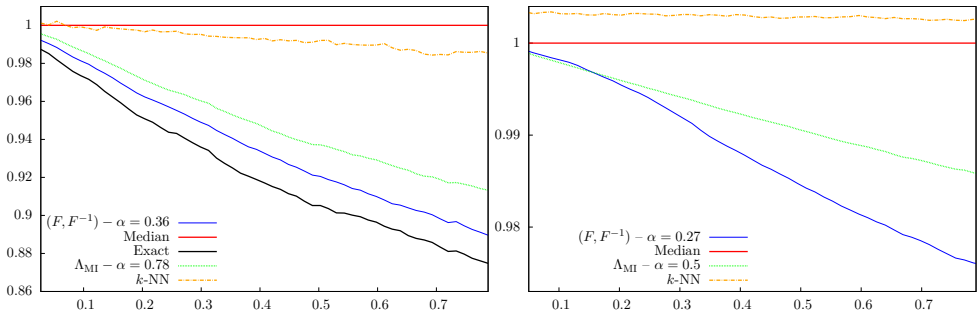


Figure 7: Mean L^1 prediction error on unobserved variables, as a function of the proportion of revealed variables, for the urban networks of Figure 6. All values are relative to the error made by the median predictor.

7.2 Some traffic reconstruction experiments.

Before showing some results on models related to road networks, let us come back to the description of the road traffic example application, alluded to in the introduction. The classical way to obtain data on a road traffic network is to install fixed sensors, such as magnetic loops. However, this is adapted to highways and arterial roads, but not to a whole urban network which typically scales up to 10^5 segments. As part of the Field Operational Test PUMAS [31] in Rouen (Normandy), we explored the possibility of acquiring data with equipped vehicles that send geolocalized information, and to process it directly with a fast prediction scheme (Furtlehner et al. [14]). This approach is related to the work of Herrera et al. [17]. The system is partially observed and the goal is to predict the complete state of the traffic network, which is represented as a high-dimensional real valued vector of travel times, speeds or densities. State of the art methods in this field exploit both temporal and spatial correlations with multivariate regression (Min and Wynter [28]), with rather restrictive linear hypotheses on the interactions. The different route we explore here for encoding spatial and potentially temporal dependencies simplifies both the model calibration and the data reconstruction tasks for large scale systems.

A simple road network model. Let us now consider two new synthetic models, associated to the road networks of Figure 6, which can be seen as very rough descriptions of city networks. The dependency graph of the vector $\mathbf{X}$ is basically the line graph of the road networks, i.e. there is a direct dependency between edges i and j iff they are adjacent in the road network. For both models the graph mean connectivity is between 5 and 6. To model the impact of a ring road on its neighborhood, we set their partial correlations with adjacent edges to 0.3. The marginal distributions

of travel times are real data coming from the Australian M4 motorway. We assume that the ring road links are always observed, by means of specific equipment such as magnetic loops. In both cases, we worked with training samples of size 10,000, which is smaller than the vector dimension (13,752) for the large model. For the network of Figure 6-right, we will not present the perfect predictor performance, since GaBP is not convergent on the true model and exact computation would involve working with matrices too large to be done in reasonable time.

The decimation experiments are performed 1,000 times for the small model (Figure 6-left) and 50 times for the larger one (Figure 6-right). The results are presented in Figure 7. Note that, in these cases, the k-NN predictor performance is very bad, due to the fact that correlations are small compared to the vector dimension (Beyer et al. [2]). The parameter α of (14) is estimated with a dichotomy search on $[0, 1]$ up to a precision of 0.01. Once again, the best choice of encoding function is the cdf, which performs clearly better than Λ^{MI}. Note that on the larger networks results are harder to interpret because of two reasons: the absence of comparison to the perfect predictor (which computation is too long) and the fact that the network contains mainly weak correlations.

To give an idea of the actual scalability of our method, the mean running time for BP to converge on the larger network is around 0.4 seconds with the choice (F, F^{-1}) and 0.2 seconds for Λ^{MI}.

Let us now turn to an experiment based on actual traffic data.

The Paris urban area highway network. As far as real data experiments are concerned, we have not yet at our disposal a dataset of urban floating car data covering a complete network, on which we could test the method in real conditions on a large scale. Nevertheless, it is by now quite easy to get data from static sensors, installed on the main axes of most highway networks and measure the local vehicle speeds (averaged over 5 minutes in our case). The dataset we consider corresponds to the *Île-de-France* urban area, between 5 am and 10 pm, and aggregated over a period of roughly four months, from March to June 2014 (excluding week-ends and day-offs). After cleaning the data, from a number of 4,638 stations, we end up with a set of 1,632 stations with an activity rate greater than 50%. Our dataset consists of $204 \times 69 = 14{,}076$ samples of the speed vector $X_t \in \mathbb{R}^{1632}$, with roughly between 10% to 20% of missing entries in average. The dataset is divided into two parts, the training set, used to build the model, of size 12,240, and a test set of size 1,836, corresponding to the last 9 days of June.

Contrary to the previous synthetic example, the graph of the inference model is not given in advance, because the stations we are using are scattered randomly over the network, and are not necessarily associated to contiguous segments. Considering nearest $2d$ spatial neighbors doesn't then make

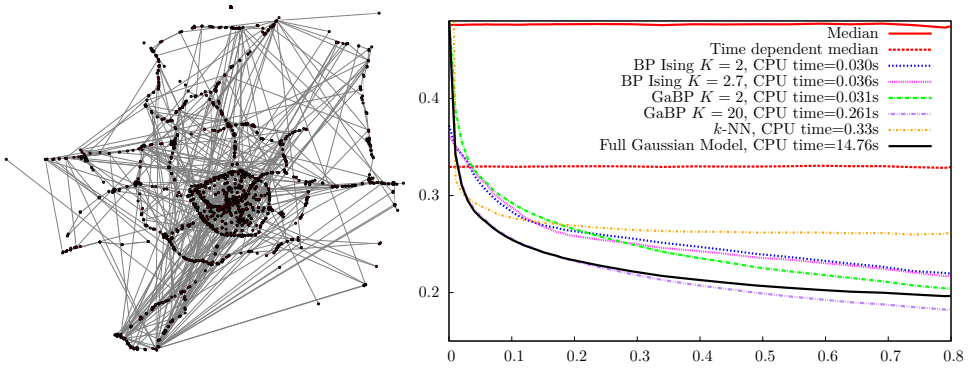


Figure 8: (left) Graph obtained with $\star$-IPS of mean connectivity 2 used by BP-Ising. (right) L^1 relative speed error as a function of the proportion of observed variables. Comparison between BP-Ising, k-NN, GaBP and exact inverse Gaussian on the decimation experiment. Each curve is drawn averaged over 1,000 runs. The indicated running times in seconds represents the mean CPU time needed to perform one inference over the network, averaged over all the runs.

much sense. In such a situation, as discussed previously in Section 4.2, no exact optimal way is available to calibrate the model, the task is complicated by the lack of a graph structure known in advance. From this viewpoint, Gaussian MRF (GMRF) based model are in a better situation, since many efficient methods exists to identify graphs corresponding to the inverse covariance matrix [12]. In fact in [25] we have developed a specific algorithm called $\star$-IPS to address this question, namely building a GMRF compatible with Gaussian belief propagation (GaBP). Since solving this problem is outside the scope of this paper, we simply use here the various graphs given by $\star$-IPS in combination with the results of Section 4.2 to calibrate the model. In fact this procedure makes sense only for very sparse graphs, close to the maximal spanning tree of mutual information [13], because then most important links are expected to be properly selected, without deteriorating too much the Bethe approximation underlying Section 4.2. An instance of a graph selected for BP-Ising is shown on Figure 8-left. We see that, many long range links are present, even though by looking at the distributions (not shown) most of the links are short range as expected. This is clearly an effect of the GMRF modeling, which cannot propagate long range correlations from short range interactions. Instead Ising models should not rely on long range interactions to propagate long range correlations, especially on dense urban traffic network, but we are not yet able to verify it properly.

Then, we perform once again the decimation experiment and the com-

parison between different models and procedures is shown on Figure 8-right. The encoding used for BP-Ising is (F_X, F_X^{-1}). Note that for the GMRF we supposed a Gaussian copula which amounts to the encoding $(F_{\mathcal{N}(0,1)}^{-1} \circ F_X, F_X^{-1} \circ F_{\mathcal{N}(0,1)})$. The best performance is obtained with the GMRF modeling, where we see that sparse models compatible with GaBP can be found and perform better and much faster than the "exact" Gaussian procedure, which seems to slightly overfit the data in this example. k-NN is very efficient to find the global traffic pattern with very few observed variables, but then it does not gain any additional information when more variables are observed, in contrary to the GMRF modeling, which is able to gain information by exploiting local correlations. Concerning BP-Ising, we observed that it has similar performances with GMRF of similar complexity, i.e. model having a mean connectivity $K \in [2, 3]$, both regarding the reconstruction error and computation time. So clearly the encoding strategy we have developed in this paper for binary variables is correctly working on this example, which it was not designed for. Efficient models with higher connectivity are not accessible yet with the ad hoc calibration procedure used for this dataset. In order to obtain a modeling which gets the best of the two worlds—GMRF for local correlations and k-NN for global patterns—which is in a way the promise of the discrete MRF modeling, we have yet to improve the procedure of model calibration.

8 Conclusion

A new method to model the interaction between real-valued random variables defined over a graph has been proposed. It relies on the following information:

- the empirical cumulative distribution function of each variable;
- a potentially incomplete covariance matrix.

This method relies on a MRF model with binary latent variables. A binary perception is built as a latent state descriptor and the dependencies between the real state variables are encoded at this latent level. Global patterns can then emerge from local interactions. This leads to a minimal parametric method, where both the prediction and the model estimation are easy to perform.

While the original motivation is traffic reconstruction, this method applies to any kind of mobile sensors moving on a large network and measuring some quantity. Sensors on shared bikes could measure air pollution for example. This is particularly true in a context of very sparse observations.

Other possible approaches. Before moving forward with the discussion of our results, it is interesting to examine our reliance on binary latent vari-

ables, since it is not self-evident. The most natural alternative would have been to build a multivariate Gaussian copula, since then the Gaussian belief propagation algorithm can be run efficiently on such models. However, worthwhile complex systems are likely to be multimodal, and a Gaussian model, which is always unimodal and admits only one reference state associated to one single belief propagation fixed point [3], might be not the most suitable modeling hypothesis. Conversely, binary models are well studied in statistical physics—under the spin glass terminology [27]—and can exhibit metastable states corresponding to different modes. Each mode can be, under certain conditions, associated to a belief propagation fixed point [15]. In a preliminary work, we have indeed observed on simulated traffic data, that belief propagation fixed points can represent the main global traffic patterns [14].

Another route to obtain multimodal models is the use of a Gaussian mixture model, where the number of components can be learned from the data. However, the BP algorithm is well defined for real-valued variables only in the case of a Gaussian vector $\mathbf{X}$. For Gaussian mixture models, the nonparametric BP algorithm has been proposed by Sudderth et al. [34], but it is computationally heavier than classical BP. Indeed, it has to store and update a number of particles for each marginal distribution and its complexity scales as particle filtering does, but applied to a larger class of PGMs. Even with the improvements proposed by Han et al. [16], this approach remains too time-consuming for real-time applicability on large networks. By contrast, the BP-based method described in this paper keeps computations lightweight, with a complexity scaling as the number of edges in the latent variables graph.

Results and discussion. Let us now come back to discuss the results of the method and possible extensions.

The choice of the cdf as encoding function and its inverse as decoding function seems to be the best one, as long as the graph connectivity is not too high. When this connectivity increases too much, the mBP algorithm looses its efficiency and one should rather choose the encoding function Λ^{MI}. A potentially difficult but sometimes necessary task, as illustrated by the Paris urban area dataset in Section 7, has been left aside in this paper: finding the dependency graph structure. This task can be performed in principle using greedy heuristics (see Jalali et al. [19]) or L^1-based regularization method (see Ravikumar et al. [32]). But some additional effort has to be put forth to make it fully efficient in our context, where compatibility with BP is mandatory. Adapting the $\star$-IPS strategy [25] to the context of discrete models is a line of research under investigation.

Nevertheless, in our target application we expect the dependency graph to be imposed by the data. Once the encoding/decoding functions are cho-

sen and the marginals p_{ij} have been estimated, many available methods exist to define the latent PGM, i.e. the set of couplings. The best one will depend on the data and determining it will require tests on real data. However, the results presented here make us quite optimistic about applying this method to urban road traffic data, for which the underlying binary description seems natural. A natural improvement would be to make the "temperature" parameter α in (14) depend on the proportion ρ of revealed variables. This makes sense since this parameter aims to avoid overcounting interactions, which is roughly related to the number of cycles, decreasing when ρ increases.

Straightforward generalization of the approach presented here can be carried out to construct latent variables with a feature space larger than $\{0, 1\}$, by considering additional random thresholds defined in Section 3.1 or deterministic ones; the underlying principles remain unchanged. In particular, it is still possible to build decoding functions based on ML or on Jeffrey's rule, to use the EM algorithm for pairwise distributions estimation and the mBP algorithm for inference.

A Pairwise model justification with Jaynes' maximum entropy principle

Let us assume that, for a given set $\mathbb{E} \subset \mathbb{V}^2$ of node pairs, we know the pairwise marginal distributions

$$\forall (i,j) \in \mathbb{E},\ \mathbb{P}(\sigma_i = s_i, \sigma_j = s_j) = p_{ij}(s_i, s_j). \tag{30}$$

We want to express the probability distribution of the vector $\boldsymbol{\sigma}$ maximizing the entropy, subject to the constraints (30). The entropy is expressed as:

$$S(\mathbb{P}) = -\sum_{\boldsymbol{\sigma}=\mathbf{s}} \mathbb{P}(\boldsymbol{\sigma} = \mathbf{s}) \log \mathbb{P}(\boldsymbol{\sigma} = \mathbf{s}).$$

The Lagrangian of the given maximization problem reads

$$\mathcal{L}(\mathbb{P}, \boldsymbol{\lambda}) = S(\mathbb{P}) + \sum_{(i,j)\in\mathbb{E}} \lambda_{ij}(s_i, s_j) \left(\sum_{\substack{\boldsymbol{\sigma}=\mathbf{s}| \\ \sigma_i = s_i, \sigma_j = s_j}} \mathbb{P}(\boldsymbol{\sigma} = \mathbf{s}) - p_{ij}(s_i, s_j) \right),$$

and its stationary points yield a distribution of the form

$$\mathbb{P}(\boldsymbol{\sigma} = \mathbf{s}) \propto \exp\left(\sum_{(i,j)\in\mathbb{E}} \lambda_{ij}(s_i, s_j) \right) = \prod_{(i,j)\in\mathbb{E}} e^{\lambda_{ij}(s_i, s_j)}. \tag{31}$$

In the end we obtain a pairwise PGM for $\boldsymbol{\sigma}$ as in (3) and thus (31) is a pairwise model. The only assumptions leading to this is that the pairwise observation (1) without any other prior cannot lead to more than pairwise marginal distributions combined with Jaynes' maximum entropy principle.

B Proof of Proposition 4

We will prove that the Kullback-Leibler divergence between the empirical joint distribution $\mathcal{P}$ of (X_i, X_j) and the joint distribution $\mathbb{P}$ within our model is strictly positive as soon as $I_{\mathcal{P}}(X_i, X_j) > \log(2)$.

$$\begin{aligned} D_{\mathrm{KL}}(\mathcal{P}\|\mathbb{P}) &= \int \mathcal{P}(x_i, x_j) \log \frac{\mathcal{P}(x_i, x_j)}{\mathbb{P}(x_i, x_j)} dx_i dx_j \\ &= I_{\mathcal{P}}(X_i, X_j) + \int \mathcal{P}(x_i, x_j) \log \frac{\mathcal{P}(x_i)\mathcal{P}(x_j)}{\mathbb{P}(x_i, x_j)} dx_i dx_j, \\ &\stackrel{\text{def}}{=} I_{\mathcal{P}}(X_i, X_j) - \mathbb{I}(X_i, X_j). \end{aligned}$$

Using the fact that $\mathbb{P}(x_i) = \mathcal{P}(x_i)$ and expanding w.r.t. σ_i and σ_j, one gets

$$\begin{aligned} \mathbb{I}(X_i, X_j) &= \int \mathcal{P}(x_i, x_j) \log \left(\sum_{\sigma_i, \sigma_j} \frac{\mathbb{P}(\sigma_i, \sigma_j)}{\mathbb{P}(\sigma_i)\mathbb{P}(\sigma_j)} \Lambda_i^{\sigma_i}(x_i) \Lambda_j^{\sigma_j}(x_j) \right) dx_i dx_j, \\ &\leq \log \left(\int \mathcal{P}(x_i, x_j) \sum_{\sigma_i, \sigma_j} \frac{\mathbb{P}(\sigma_i, \sigma_j)}{\mathbb{P}(\sigma_i)\mathbb{P}(\sigma_j)} \Lambda_i^{\sigma_i}(x_i) \Lambda_j^{\sigma_j}(x_j) dx_i dx_j \right), \end{aligned}$$

with $\Lambda^1 \stackrel{\text{def}}{=} \Lambda$ and $\Lambda^0 \stackrel{\text{def}}{=} 1 - \Lambda$. Defining $\mathcal{P}_{\sigma_i\sigma_j} \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{P}}[\Lambda_i^{\sigma_i}(X_i)\Lambda_j^{\sigma_j}(X_j)]$, we get the final expression

$$\mathbb{I}(X_i, X_j) \leq \log \left(\sum_{\sigma_i, \sigma_j} \frac{\mathbb{P}(\sigma_i, \sigma_j)}{\mathbb{P}(\sigma_i)\mathbb{P}(\sigma_j)} \mathcal{P}_{\sigma_i\sigma_j} \right) \leq \log(2),$$

because we have $\mathbb{P}(\sigma_i, \sigma_j) \leq \mathbb{P}(\sigma_j)$ and $\sum_{\sigma_j} \mathcal{P}_{\sigma_i\sigma_j} = \mathbb{P}(\sigma_i)$.

C Proof of Proposition 5

Let us focus first on the case of one factor with two binary variables σ_i and σ_j, both observed (Figure 9 with $n = 2$). The messages $m_{a\to i}$ are assumed to be normalized such that

$$\sum_{s_i} m_{a\to i}(s_i) = 1.$$

We introduce the following notation

$$u_n \stackrel{\text{def}}{=} m_{a\to i}(0), \quad v_n \stackrel{\text{def}}{=} m_{a\to j}(0),$$

so that $1-u_n = m_{a\to i}(1)$ and $1-v_n = m_{a\to j}(1)$. Using the update rules (16)–(17), one obtains:

$$u_{n+1} = \frac{\psi_{00}\alpha_j\bar{v}_n + \psi_{01}\bar{\alpha}_j v_n}{(\psi_{00}+\psi_{10})\alpha_j\bar{v}_n + (\psi_{01}+\psi_{11})\bar{\alpha}_j v_n}, \tag{32}$$

$$v_{n+1} = \frac{\psi_{00}\alpha_i\bar{u}_n + \psi_{10}\bar{\alpha}_i u_n}{(\psi_{00}+\psi_{01})\alpha_i\bar{u}_n + (\psi_{10}+\psi_{11})\bar{\alpha}_i u_n}, \tag{33}$$

where $\alpha_i \stackrel{\text{def}}{=} b_i^*(0)$, $\psi_{yz} \stackrel{\text{def}}{=} \psi(\sigma_i = y, \sigma_j = z)$ and using the convention $\bar{z} \stackrel{\text{def}}{=} 1 - z$.

Lemma 1. *The sequences $(u_n)_{n\in\mathbb{N}}$ and $(v_n)_{n\in\mathbb{N}}$, defined recursively by (32) and (33), converge to a unique fixed point for any $(u_0, v_0) \in]0,1[^2$.*

Proof. Since the roles of u_n or v_n are symmetric, we will only prove the convergence of u_n. From (32) and (33), we obtain a recursive equation of the form $u_{n+2} = f(u_n)$ such as

$$f(x) = \frac{h_0 x + K_0}{(h_0 + h_1)x + (K_0 + K_1)},$$

with

$$\begin{aligned}
h_0 &\stackrel{\text{def}}{=} \psi_{00}\alpha_j(\bar{\alpha}_i\psi_{11} - \alpha_i\psi_{01}) + \psi_{01}\bar{\alpha}_j(\psi_{10}\bar{\alpha}_i - \psi_{00}\alpha_i),\\
h_1 &\stackrel{\text{def}}{=} \psi_{10}\alpha_j(\bar{\alpha}_i\psi_{11} - \alpha_i\psi_{01}) + \psi_{11}\bar{\alpha}_j(\psi_{10}\bar{\alpha}_i - \psi_{00}\alpha_i),\\
K_0 &\stackrel{\text{def}}{=} \psi_{00}\psi_{01}\alpha_i, \quad K_1 \stackrel{\text{def}}{=} \alpha_i(\psi_{10}\psi_{01}\alpha_j + \psi_{11}\psi_{00}\bar{\alpha}_j).
\end{aligned}$$

The derivative of f is

$$f'(x) = \frac{h_0K_1 - h_1K_0}{\big((h_0 + h_1)x + (K_0 + K_1)\big)^2},$$

which is of constant sign. If $f'(x) \geq 0$, then u_{2n} and u_{2n+1} are monotonic, and, because u_n is bounded, we can conclude that both u_{2n} and u_{2n+1} converge. If we could prove that there is a unique fixed point in the interval $[0,1]$, we would have proved that u_n converges.

Let us begin by discarding some trivial cases. First the case $f(1) = 1$ implies that $\alpha_i = 1$ and

$$\begin{aligned}
h_0 &= -\psi_{00}\psi_{01} = -K_0,\\
h_1 &= -\psi_{10}\psi_{01}\alpha_j - \psi_{11}\psi_{00}\bar{\alpha}_j = -K_1,
\end{aligned}$$

which leads to f being a constant function equal to $\frac{K_0}{K_0+K_1}$. When $f(0) = 0$, one has $\alpha_i = K_0 = K_1 = 0$, and f is again constant. The cases $f(1) = 0$ and $f(0) = 1$ are treated similarly and f is still a constant function, which implies the trivial convergence of u_n.

Case 1: f is increasing At least one fixed point exists in $[0,1]$ since $f([0,1]) \subset [0,1]$. Studying the roots of $f(x) - x$ shows that the number of fixed points is at most 2 since these fixed points are roots of a degree 2 polynomial.

Since $f(0) > 0$, f being increasing and $f(1) < 1$ the number of fixed points has to be odd, indeed the graph of f must cross an odd number of times the first bisector. One can conclude that there is only one fixed point in $[0,1]$, so both u_{2n} and u_{2n+1} converge to the same fixed point.

Figure 9: Chain of N pairwise factors, the extremal variables σ_1 and σ_N are observed.

Case 2: f is decreasing We just have to consider the sequence $(1 - u_n)_{n\in\mathbb{N}}$, which is similar, but will be defined by recurrence of the form $1 - u_{n+2} = g(1 - u_n)$ with a function g such as g' is positive and the result of Case 1 applies. ■

The case we just studied is in fact much more general than it looks. Indeed, as soon as a tree gets stuck between exactly two nodes with fixed beliefs, the situation is equivalent and leads to the result of Proposition 5.

Proof of Proposition 5. First it is trivial to see that fixing the beliefs of a set of nodes $\mathbb{V}^* \subset \mathbb{V}$ has the effect of the transformation $\mathcal{T}(\cdot, \mathbb{V}^*)$ in term of messages propagation. To conclude the proof, it is enough to focus on proving the convergence on a tree with two leaves in $\mathbb{V}^*$. Consider the tree of Figure 9; one can show that it is equivalent to the case of Lemma 1 for a well chosen function ψ. Propagating the updates rules yields $m_{a_1\to 1}(s_1) \leftarrow \Theta$, with

$$\Theta \propto \sum_{s_N} \Big(\sum_{s_1 \dots s_{N-2}} \prod_{i=1}^{N-2} \psi_{a_i}(\mathbf{s}_{a_i}) \phi_i(s_i) \Big) \frac{\psi_{a_{N-1}}(\mathbf{s}_{a_{N-1}}) b_r^*(s_N)}{m_{a_{N-1}\to N}(s_N)}.$$

We define $\tilde{\psi}$ such as

$$\tilde{\psi}(s_1, s_N) = \Big(\sum_{s_1 \dots s_{N-2}} \prod_{i=1}^{N-2} \psi_{a_i}(\mathbf{s}_{a_i}) \phi_i(s_i) \Big) \ \psi_{a_{N-1}}(\mathbf{s}_{a_{N-1}}),$$

then we use the results of Lemma 1 to obtain the convergence of messages on this tree. In the general case of a tree with two leaves in $\mathbb{V}^*$, leaves fixed on variables $\sigma_i, i \in \{1 \dots N\}$ will simply affect the local fields ϕ_i. The leaves fixed on factors a_i will affect the functions ψ_{a_i}. In fact, since the graph is a tree, we know that the information sent by σ_1 and σ_N to these leaves will not come back to σ_1 and σ_N. These leaves send constant messages, which can be integrated into the functions ψ and ϕ, in order to recover the setting of Lemma 1. ■

References

[1] R. Baxter. *Exactly Solved Models in Statistical Mechanics.* Dover Publications, 2008.

[2] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Database Theory-ICDT'99*, pages 217–235. Springer, 1999.

[3] D. Bickson. *Gaussian Belief Propagation: Theory and Application*. PhD thesis, Hebrew University of Jerusalem, 2008.

[4] J. Bilmes. On soft evidence in Bayesian networks. Technical report, University of Washington, 2004.

[5] X. Boyen. *Inference and Learning in Complex Stochastic Processes*. PhD thesis, Stanford University, Computer Science Department, 2002. 229 pages.

[6] H. Chan and A. Darwiche. On the revision of probabilistic beliefs using uncertain evidence. *Artificial Intelligence*, 163(1):67–90, 2005.

[7] S. Cocco and R. Monasson. Adaptive cluster expansion for the inverse Ising problem: convergence, algorithm and tests. *Journal of Statistical Physics*, 147(2):252–314, 2012.

[8] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[9] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[11] A. Doucet, N. de Freitas, and N. Gordon. *An Introduction to Sequential Monte Carlo Methods*. New York: Springer-Verlag, 2001.

[12] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[13] C. Furtlehner. Approximate inverse Ising models close to a Bethe reference point. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(09):P09020, 2013.

[14] C. Furtlehner, Y. Han, J.-M. Lasgouttes, V. Martin, F. Marchal, and F. Moutarde. Spatial and temporal analysis of traffic states on large scale networks. In *Procceding of the 13th International IEEE Conference on Intelligent Transportation Systems*, pages 1215–1220, 2010.

[15] C. Furtlehner, J.-M. Lasgouttes, and A. Auger. Learning multiple belief propagation fixed points for real time inference. *Physica A: Statistical Mechanics and its Applications*, 389(1):149–163, 2010.

[16] T. X. Han, H. Ning, and T. S. Huang. Efficient nonparametric belief propagation with application to articulated body tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 214–221, 2006.

[17] J. Herrera, D. Work, R. Herring, X. Ban, Q. Jacobson, and A. Bayen. Evaluation of traffic data obtained via GPS-enabled mobile phones: The mobile century field experiment. *Transportation Research Part C: Emerging Technologies*, 18(4):568–583, 2010.

[18] A. Ihler, J. I. Fischer, and A. Willsky. Loopy belief propagation: convergence and effects of message errors. *The Journal of Machine Learning Research*, 6:905–936, 2005.

[19] A. Jalali, C. Johnson, and P. Ravikumar. On learning discrete graphical models using greedy methods. *arXiv preprint arXiv:1107.3258*, 2011.

[20] E. T. Jaynes. Prior probabilities. *Systems Science and Cybernetics, IEEE Transactions on*, 4(3):227–241, 1968.

[21] E. T. Jaynes. *Probability Theory: The Logic of Science (Vol 1)*. Cambridge University Press, 2003. ISBN 0521592712.

[22] F. R. Kschischang, B. J. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[23] D. J. Mackay, J. S. Yedidia, W. T. Freeman, Y. Weiss, et al. A conversation about the Bethe free energy and sum-product. Available at `http://www.merl.com/publications/TR2001-018/`, 2001.

[24] V. Martin. *Modélisation Probabiliste et inférence par l'algorithme belief propagation*. PhD thesis, Mines-ParisTech, 2013.

[25] V. Martin, C. Furtlehner, Y. Han, and J.-M. Lasgouttes. GMRF Estimation under Spectral and Topological constraints. In *Machine Learning and Knowledge Discovery in Databases*, volume 8725 of *Lecture Notes in Computer Science*, pages 370–385. Springer Berlin Heidelberg, 2014.

[26] M. Mézard and T. Mora. Constraint satisfaction problems and neural networks: A statistical physics perspective. *Journal of Physiology-Paris*, 103(1-2):107–113, 2009.

[27] M. Mézard, G. Parisi, and M. Virasoro. *Spin Glass Theory and Beyond*. World Scientific, Singapore, 1987.

[28] W. Min and L. Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C*, 19:606–616, 2011.

[29] J. M. Mooij and H. J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. *Information Theory, IEEE Transactions on*, 53(12):4422–4437, 2007.

[30] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference*. Morgan Kaufmann, 1988.

[31] PUMAS project. `http://team.inria.fr/pumas/` (in French).

[32] P. Ravikumar, M. J. Wainwright, and J. D. Lafferty. High-dimensional Ising model selection using L^1 regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.

[33] B. L. Smith, B. M. Williams, and R. Keith Oswald. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C: Emerging Technologies*, 10(4):303–321, 2002.

[34] E. Sudderth, A. Ihler, M. Isard, W. Freeman, and A. Willsky. Nonparametric Belief Propagation. *Communications of the ACM*, 53(10): 95–103, Oct. 2010.

[35] S. Tatikonda and M. Jordan. Loopy Belief Propagation and Gibbs measures. In *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence*, pages 493–50, 2002.

[36] Y. W. Teh and M. Welling. Passing and bouncing messages for generalized inference. Technical report, UCL, 2001.

[37] M. J. Wainwright. Estimating the "wrong" graphical model: benefits in the computation-limited setting. *The Journal of Machine Learning Research*, 7:1829–1859, 2006.

[38] M. Welling and Y. W. Teh. Approximate inference in Boltzmann machines. *Artificial Intelligence*, 143(1):19–50, 2003.

[39] M. Yasuda and K. Tanaka. Approximate learning algorithm in Boltzmann machines. *Neural computation*, 21(11):3130–3178, 2009.

[40] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized Belief Propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.