Particle Methods as Message Passing

Justin Dauwels
RIKEN Brain Science Institute
Hirosawa, 2-1, Wako-shi, Saitama, Japan
Email: justin@dauwels.com

Sascha Korl
Phonak AG
CH-8712 Staefa, Switzerland
Email: sascha.korl@phonak.ch

Hans-Andrea Loeliger
Dept. of Information Technology and Electr. Eng.,
ETH Zurich, CH-8092 Zurich, Switzerland
Email: loeliger@isi.ee.ethz.ch

***Abstract*— It is shown how particle methods can be viewed as message passing on factor graphs. In this setting, particle methods can readily be combined with other message-passing techniques such as the sum-product and max-product algorithm, expectation maximization, iterative conditional modes, steepest descent, Kalman filters, etc. Generic message computation rules for particle-based representations of sum-product messages are formulated. Various existing particle methods are described as instances of those generic rules, i.e., Gibbs sampling, importance sampling, Markov-chain Monte Carlo methods (MCMC), particle filtering, and simulated annealing.**

I. Introduction

Particle methods (or "Monte-Carlo methods") have in the last fifty years intensively been used to solve a wide variety of problems in physics and computer science (e.g., statistical inference and optimization). The main idea behind particle methods is to represent a probability density function (pdf) or probability mass function (pmf) as a list of mass points ("particles"), as illustrated in Fig. 1. A (positive) weight is associated to each particle in the list; the weights of all particles are supposed to add to 1. If the mass points are samples from the pdf or pmf at hand, all weights are identical ("uniform") (see Fig. 1 (left)); if the particles are generated by other means (e.g., they may be sampled from a different function than f), non-uniform weights are associated to the particles (see Fig. 1 (right)). A list $\mathcal{L}_f$ of N particles representing the pdf or pmf $f(x)$ with $x \in \mathcal{X}$ is thus formally defined as a list of pairs

$$\mathcal{L}_f \triangleq \big\{(\hat{x}^{(1)}, w^{(1)}), (\hat{x}^{(2)}, w^{(2)}), \ldots, (\hat{x}^{(N)}, w^{(N)})\big\},$$

where $\hat{x}^{(i)} \in \mathcal{X}$, the weights $w^{(i)}$ are positive real numbers and $\sum_i w^{(i)} = 1$.

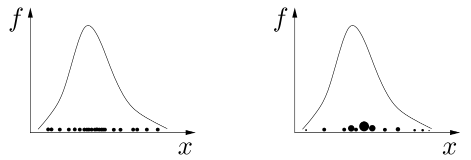

Fig. 1. A probability density function $f : \mathbb{R} \to \mathbb{R}^+$ and its representation as a list of particles. The radius of the particles is proportional to their weights. (left) uniform weights; (right) non-uniform weights.

Particle methods can be used to evaluate the expectation of some function g w.r.t. a pdf f, i.e.,

$$\mathrm{E}_f[g] \triangleq \int_x g(x)f(x)dx, \tag{1}$$

where $\int_x h(x)dx$ stands either for integration or summation of h over the whole range of X. The expression (1) is ubiquitous in statistical inference; for example, the minimum mean square error estimate (MMSE) of a real random variable (or vector) X from an observation $Y = y$ is

$$\hat{x}_{\text{MMSE}} \triangleq \mathrm{E}_{X|Y}[X] \triangleq \int_x x f(x|y)dx, \tag{2}$$

where $f(x|y)$ is the posterior pdf. If the expression (1) is intractable, one needs to resort to approximations; in particle methods, the expectation (1) is approximated as an average over a particle list $\mathcal{L}_f$:

$$\mathrm{E}_f[g] \approx \sum_{i=1}^{N} w^{(i)} g(\hat{x}^{(i)}). \tag{3}$$

In many practical problems, the pdf f and the function g have a "nice" structure, i.e., they factorize; particle lists of f can then be generated by means of simple local computations; also the evaluation of (3) then only involves local computations. We will show in this paper that particle methods can be viewed as message-passing algorithms operating on factor graphs where messages are represented as lists of samples [1] [2]. In other words, particle methods are a full member of the family of message-passing methods; this enables us for example:

- to apply particle methods *locally*, i.e., at a particular node in the factor graph of the system at hand;
- to combine particle methods with other message-passing techniques such as the sum-product and max-product algorithm [1], expectation maximization [3], iterative conditional modes [4], steepest descent [5], Kalman filters [6], etc.;
- to apply particle methods on factor graphs with cycles;
- to choose the order in which particle lists ("messages") are updated ("message-update schedule").

The use of particle lists in message passing algorithms was proposed in [7], [1], [2]. In this paper, we go into more detail. In particular:

- We investigate how particle methods can be used to (approximately) compute *marginals* when straightforward sum-product message passing is intractable. We formulate generic message computation rules for particle-based representations of sum-product messages; we thereby systematically investigate various types of incoming messages.

- We describe various existing methods for drawing *samples* from a multivariate pdf as message passing on factor graphs.

This paper is structured as follows. In the following section, we explain how the sum-product message computation rule may be approximated by particle methods; particle-based representations of sum-product messages are thereby required. We investigate in Section III how such representations can be obtained. In Section IV, we present various existing particle methods as message passing on factor graphs.

II. Approximating the sum-product rule by particle methods

Suppose that we wish to compute marginals from a given multivariate function $f(x_1, x_2, \ldots, x_n)$, where the variables X_k may be discrete or continuous. As is well known, marginals of a function f can in principle be computed by applying the sum-product algorithm on a cycle-free factor graph of f [1]. However, this naive approach fails if the sum-product rule is intractable at some node(s) in the factor graph of f, which is often the case if (some of) the variables X_k are continuous. In the following, we take a closer look at this issue. A generic node h in the factor graph of f is depicted in Fig. 2.

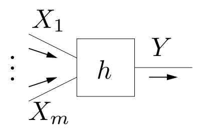

Fig. 2. Message along a generic edge.

The sum-product message towards edge Y is given by [1]:

$$\mu_Y(y) \overset{\triangle}{\propto} \int_{x_1,\ldots,x_m} h(y, x_1, \ldots, x_m)\mu_{X_1}(x_1)\cdots\mu_{X_m}(x_m)dx, \quad (4)$$

where μ_{X_k} is the message that arrives at node h along the edge X_k (see Fig. 2). The expression (4) may be intractable for the following reasons.

- If the variables X_k in Fig. 2 are discrete, the expression (4) can in principle be evaluated straightforwardly. However, the summation occurring in (4) can only be carried out in practice if the alphabets of the variables X_k are sufficiently small.
- If (some of) the variables X_k are continuous, the expression (4) may lead to intractable integrals.

If the expression (4) is infeasible for one of the above reasons (or both), one needs to resort to approximative methods such as particle methods. Suppose, without loss of generality, that the summation/integration over X_1 in (4) is infeasible. This summation/integration may be approximated by means of particle methods as follows: the message μ_{X_1} is represented as a *particle list* and the rule (4) is evaluated as (cf. (3)):

$$\mu_Y(y) \overset{\triangle}{\propto} \sum_i \int_{x_2,\ldots,x_m} h(y, \hat{x}_1^{(i)}, x_2, \ldots, x_m) \cdot w_1^{(i)} \cdot \mu_{X_2}(x_2)\cdots\mu_{X_m}(x_m)d\mu, \quad (5)$$

where $\hat{x}_1^{(i)}$ is the i-th particle in the particle list that represents the sum-product message μ_{X_1} and $w_1^{(i)}$ is the weight associated to $\hat{x}_1^{(i)}$.

Particle methods are not the only option to handle intractable sum-product message computation rule (see e.g., [1] [2]). If X_1 is continuous, one may alternatively represent the message μ_{X_1} as:

- a quantized-variable message
- a Gaussian distribution (as in Kalman filtering [6])
- a single value $\hat{x}_1$, e.g., the mode or mean of μ_{X_1} (as in decision-directed algorithms).

Obviously, if in (4) also the summation/integration over some other variable(s) X_k is intractable, one may also choose one of the above representations for the corresponding message(s) μ_{X_k}; each of those messages may be represented differently. In other words, at every node in the factor graph of the system at hand, one has the freedom to combine various types of messages, and hence, various types of algorithms, e.g., decision-based algorithms (such as iterative conditional modes and expectation maximization), Kalman filters, and particle methods. Following this approach, we have derived various message-passing algorithms for code-aided phase estimation [4] and estimation in AR models [8].

III. Generating a particle list from a sum-product message

We pointed out in the previous section that if the sum-product message computation rule is intractable, it may be evaluated (approximately) by particle methods. Incoming sum-product messages are then represented as particle lists (e.g., μ_{X_1} in (5)). In this section, we describe methods to generate particle lists from sum-product messages. We again consider the generic node h depicted in Fig. 2; we wish to represent the sum-product message μ_Y towards edge Y as a particle list. First we consider the case where all variables X_k are discrete, then we will assume that all variables X_k are continuous. It is straightforward to extend our considerations to the case where some of the variables X_k are discrete and others are continuous.

A. Discrete variables X_k

Suppose that all variables X_k are discrete; in addition, suppose that the alphabets of the variables X_k are sufficiently small so that the messages μ_{X_k} can be represented by a list of their values. (If a message μ_{X_k} can not be represented in this fashion since the alphabet of X_k is too large, one may represent μ_{X_k} as a particle list; we consider this type of representation for the messages μ_{X_k} in section III-B.) One may generate samples from μ_Y by the following procedure:

1) For each $k = 1, \ldots, m$, select a value $\hat{x}_k$ of X_k with probability proportional to $\mu_{X_k}(\hat{x}_k)$.
2) Draw a sample $\hat{y}$ from $h(y, \hat{x}_1, \ldots, \hat{x}_m)$.
3) Iterate 1–2 until a sufficient number of samples from μ_Y are obtained.

The resulting particles have uniform weights; the above sampling method (Step 1–3) is therefore referred to as unweighted sampling.

Alternatively, one may draw samples $\hat{y}$ from $h(y, \hat{x}_1, \ldots, \hat{x}_m)$ for each possible m-tuple $(\hat{x}_1, \ldots, \hat{x}_m)$. The weight w associated to sample $\hat{y}$ is proportional to

$$w \propto \prod_{k=1}^{m} \mu_{X_k}(\hat{x}_k). \tag{6}$$

This sampling method is called weighted sampling since it leads to a particle list with non-uniform weights.

If one successively draws particles $\hat{x}$ from that list with probability proportional to their weights w (6), one obtains a list of particles with uniform weights; note that a particle with a large weight w (6) may be drawn several times, whereas a particle with a small weight may not be drawn at all. This technique to generate particles lists with uniform weights from particle lists with non-uniform weights is called resampling [9].

B. *Continuous variables X_k*

If the variables X_k are continuous, we distinguish the following cases:

- If a suitable closed-form expression for the message μ_Y is available, one may sample from that expression μ_Y.
- If the messages μ_{X_k} are quantized-variable messages [2], one proceeds as in the case of discrete variables X_k.
- If the messages μ_{X_k} are lists of samples, the procedure is again similar to the one for discrete variables X_k. The first step in the unweighted sampling procedure is slightly modified: for each k, one draws a particle $\hat{x}_k$ with probability proportional to its weight w_k. In weighted sampling, one draws a sample $\hat{y}$ from $h(y, \hat{x}_1, \ldots, \hat{x}_N)$ for each m-tuple of particles $(\hat{x}_1, \ldots, \hat{x}_m)$. The weight w associated to the sample $\hat{y}$ is proportional to
$$w \propto \prod_{k=1}^{m} w_k, \tag{7}$$
where w_k is the weight of particle $\hat{x}_k$.
- If the incoming messages μ_{X_k} are represented as Gaussian distributions (and no suitable closed-form expression for μ_Y is available), one may first represent the messages μ_{X_k} as particle lists and proceed as in the previous case.
- If the incoming messages are single values $\hat{x}_k$, one draws samples from $h(y, \hat{x}_1, \ldots, \hat{x}_m)$.
- The extension to the situation where the incoming messages are not all of the same type is straightforward, except if the node h is an equality constraint node (see Section III-C).

C. *Specific node functions*

So far, we have considered generic node functions h. We now investigate two important types of node functions in more detail: (i) nodes corresponding to deterministic mappings; (ii) equality constraint nodes.

- Deterministic mapping.
Suppose that the node h corresponds to a deterministic mapping $y = v(x_1, \ldots, x_m)$:
$$h(y, x_1, \ldots, x_m) \triangleq \delta(y - v(x_1, \ldots, x_m)). \tag{8}$$
Samples $\hat{y}$ from $h(y, \hat{x}_1, \ldots, \hat{x}_m)$ are then trivially obtained as $\hat{y} = v(\hat{x}_1, \ldots, \hat{x}_m)$.
- Equality constraint node.
Suppose that the node h is an equality constraint node, i.e.,
$$h(y, x_1, \ldots, x_m) \triangleq \delta(y - x_1) \prod_{k=1}^{m-1} \delta(x_{k+1} - x_k). \tag{9}$$
The message μ_Y is then given by:
$$\mu_Y(y) \propto \prod_{k=1}^{m} \mu_{X_k}(y). \tag{10}$$
We distinguish the following cases:
  - If Y is discrete, one obtains a particle list for μ_Y by selecting values $\hat{y}$ with probability proportional to $\prod_{k=1}^{m} \mu_{X_k}(\hat{y})$. This approach is only applicable if the alphabet of Y is sufficiently small. Otherwise one may represent the messages μ_{X_k} as particle lists.
  - If the messages μ_{X_k} are represented as particle lists, it is not straightforward to draw samples from (10). One may first generate a continuous representation such as mixtures of Gaussian distributions for each of the incoming messages μ_{X_k}. Efficient methods have been devised to draw samples from products of Gaussian mixtures [7] [10].
  - If Y is continuous, and a closed-form expression for the message μ_Y is available, one may obviously sample from that expression μ_Y.
  - If Y is continuous, and closed-form expressions for the incoming messages μ_{X_k} are available, but not for μ_Y, one may sample from the product (10). This may be done by importance sampling (see Section IV-B).
  - If the messages μ_{X_k} are quantized-variable messages, one proceeds as in the discrete case.
  - Also certain mixtures of the previous situations can be handled, as we will illustrate in Section IV-B by the example of importance sampling. However, an in-depth treatment of this issue goes beyond the scope of this paper.

IV. Existing particle methods viewed as message passing

We now describe several standard Monte-Carlo techniques as instances of the above generic rules, i.e.,

- Gibbs sampling,
- importance sampling,
- particle filtering ("sequential Monte-Carlo filtering"),
- Markov-Chain Monte-Carlo methods (MCMC),
- simulated annealing.

A. Gibbs Sampling

Suppose that we wish to draw samples from a multivariate probability function $f(x_1, x_2, \ldots, x_n)$. This can be done by the following iterative algorithm known as Gibbs sampling [11, p. 371–407]:

1) Choose an initial value $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$.
2) Choose an index k.
3) Draw a sample $\hat{x}_k$ from

$$f(x_k) \triangleq \frac{f(\hat{x}_1, \ldots, \hat{x}_{k-1}, x_k, \hat{x}_{k+1}, \ldots, \hat{x}_n)}{\sum_{x_k} f(\hat{x}_1, \ldots, \hat{x}_{k-1}, x_k, \hat{x}_{k+1}, \ldots, \hat{x}_n)}. \tag{11}$$

4) Iterate 2–3 a "large" number of times.

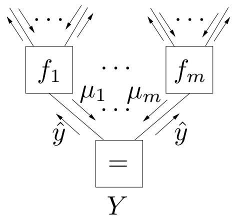

Fig. 3. Gibbs sampling at a generic edge Y.

Gibbs sampling can be interpreted as a message-passing algorithm that iterates the following steps:

1) Select a variable (equality constraint node) Y in the factor graph of f (see Fig. 3).
2) The equality constraint node Y generates the message $\hat{y}$ by sampling from:

$$f(y) \triangleq \frac{\mu_1(y) \cdot \mu_2(y) \ldots \cdot \mu_m(y)}{\sum_y \mu_1(y) \cdot \mu_2(y) \ldots \cdot \mu_m(y)}, \tag{12}$$

3) The equality constraint node broadcasts the message $\hat{y}$ to its neighboring nodes f_k $(k = 1, \ldots, m)$.

In summary: Gibbs sampling can be regarded as message passing on a factor graph, where messages are represented by a single sample.

B. Importance Sampling

Suppose that we wish to approximate the expectation (1) by some particle method. If it is "easy" to sample from f, one may draw samples $\hat{x}^{(1)}, \ldots, \hat{x}^{(N)}$ from f and evaluate the expectation (1) as

$$\mathrm{E}_f[g] \triangleq \frac{1}{N} \sum_{i=1}^{N} g\big(\hat{x}^{(i)}\big). \tag{13}$$

Suppose now that sampling from f is "hard", and hence the approach (13) is not feasible. One may then draw samples $\hat{x}^{(1)}, \ldots, \hat{x}^{(N)}$ from a (different) function $\tilde{f}$ with supp$(f) \subseteq$ supp$(\tilde{f})$, and approximate (1) as

$$\mathrm{E}_f[g] \approx \frac{1}{N} \sum_{i=1}^{N} w^{(i)} g\big(\hat{x}^{(i)}\big), \tag{14}$$

where the weights $w^{(i)}$ are given by

$$w^{(i)} \triangleq \frac{f\big(\hat{x}^{(i)}\big)}{\tilde{f}\big(\hat{x}^{(i)}\big)}. \tag{15}$$

The approach (14)–(15) is called importance sampling [11, p. 90–107]. Importance sampling is particularly natural when f factorizes. Suppose for example that

$$f(x) \triangleq f_1(x) f_2(x). \tag{16}$$

One may draw samples $\hat{x}^{(i)}$ from f_1 and weight those samples by the function f_2:

$$w^{(i)} \propto f_2\big(\hat{x}^{(i)}\big). \tag{17}$$

A message-passing view of this procedure is suggested

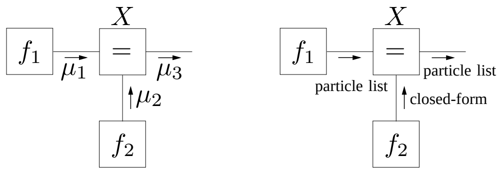

Fig. 4. Importance sampling as message passing.

in Fig. 4. The message μ_1 is a list of samples $\hat{x}^{(1)}, \ldots, \hat{x}^{(N)}$ drawn from f_1. The message μ_2 is given in closed-form, i.e., $\mu_2 \triangleq f_2$. The message μ_3 is the particle list $\mu_3 \triangleq \big\{(\hat{x}^{(i)}, w^{(i)})\big\}_{i=1}^{N}$, where $w^{(i)}$ is defined in (17). Importance sampling may be viewed as a particular instance of weighted sampling, where (1) the local node function g is an equality constraint node; (2) one of the messages is a list of samples; (3) the other message is available in closed-form.

If it is hard to draw samples from both f_1 and f_2, one may use importance sampling (14) (15) in order to obtain a particle list from either f_1 or f_2. Suppose that we generate a particle list $\big\{(\hat{x}^{(i)}, \tilde{w}^{(i)})\big\}_{i=1}^{N}$ from f_1, i.e., $\mu_1 \triangleq \big\{(\hat{x}^{(i)}, \tilde{w}^{(i)})\big\}_{i=1}^{N}$. The message μ_3 is then represented as a list of samples $\big\{\hat{x}^{(i)}, w^{(i)}\big\}_{i=1}^{N}$, where

$$w^{(i)} \propto \tilde{w}^{(i)} f_2\big(\hat{x}^{(i)}\big). \tag{18}$$

This method is the key to (standard) particle filtering, the subject of next subsection.

C. Particle Filtering

Particle filtering [9] (or "sequential Monte-Carlo integration") is a particle method for filtering in state-space models. It can be viewed as for forward-only message passing in a state-space model of the form:

$$\begin{aligned} &f(s_0, s_2, \ldots, s_n, y_1, y_2, \ldots, y_n) \\ &\qquad \triangleq f_A(s_0) \prod_{k=1}^{n} f_A(s_{k-1}, s_k) f_B(s_k, y_k), \end{aligned} \tag{19}$$

where (some of the) messages are represented by lists of samples (see Fig. 5; the figure shows only one section of the factor graph). More precisely, the messages μ_k and $\tilde{\mu}_k$ (for

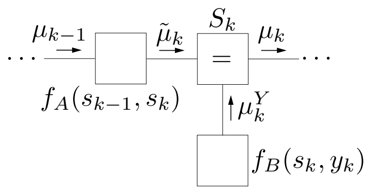

Fig. 5. Particle filtering as message passing.

all k) are represented as lists of samples. In the basic particle filter, the list $\tilde{\mu}_k$ is obtained from μ_{k-1} by weighted sampling. The sampling-importance-resampling particle filter (SIR) uses unweighted sampling instead. In both particle filters, the list μ_k is generated from $\tilde{\mu}_k$ by importance sampling (cf. Fig. 4): the message $\tilde{\mu}_k$, μ_k^Y and μ_k in Fig. 5 correspond to the message μ_1, μ_2 and μ_3 respectively in Fig. 4.

D. Markov-Chain Monte-Carlo Methods (MCMC)

Markov-Chain Monte-Carlo methods [11] (MCMC) are an alternative family of methods to draw samples from a probability function ("message") f from which it is hard to sample directly. The main idea is to sample repeatedly from an ergodic Markov chain with stationary distribution f. In the following, we briefly present the most well-known MCMC method, i.e., the Metropolis-Hastings algorithm [11]. This algorithm is based on a conditional density $q(y|x)$ from which it is assumed to be easy to sample. In addition, q is supposed to be symmetric, i.e., $q(y|x) = q(x|y)$. Usually, the function q fully factorizes, i.e.,

$$q(y_1, \ldots, y_N | x_1, \ldots, x_N) \triangleq \prod_{k=1}^{N} q(y_k|x_k). \tag{20}$$

For instance, q may be a Gaussian distribution with mean x and diagonal covariance matrix. The Metropolis-Hastings algorithm generates samples $\hat{x}$ from the "target" function f by the following iterative procedure:

1) Choose an initial value $\hat{x}$.
2) Sample $\hat{y}$ from $q(y|\hat{x})$.
3) Set $\hat{x} \triangleq \hat{y}$ with probability p where

$$p \triangleq \min\left\{\frac{f(\hat{y})}{f(\hat{x})}, 1\right\} \tag{21}$$

4) Iterate 2–3 a sufficient number of times.

Note that the function f must be available up to some constant.

Similarly as Gibbs sampling, the Metropolis-Hastings algorithm may be interpreted as a message-passing algorithm that operates on a factor graph of f. We refer to [4] for more details.

E. Simulated Annealing

The original simulated annealing algorithm is an extension of the Metropolis-Hastings algorithm [11, pp. 163–169]. It can be used (i) to sample from a multivariate function $f(x_1, \ldots, x_N)$; (ii) to find the mode of f. The key idea is to draw samples from f^α, where the (positive) exponent α slowly *increases* in the course of the algorithm. The initial value of α is close to zero (e.g., $\alpha = 0.1$). If one wishes to obtain samples from f, one halts the algorithm as soon as $\alpha = 1$. If one tries to find the mode of α, the end value of α is much larger (e.g., $\alpha = 10$ or 100). Note that for small values of α (i.e., $0 \leq \alpha < 1$), the function f^α is flatter than the target function f, whereas for large values of α (i.e., $\alpha \gg 1$), the function f^α mainly consists a narrow peak centered at the global maximum of f.

Simulated annealing works as follows:

1) Choose an initial value $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N)$.
2) Choose an initial value α (e.g., $\alpha = 0.1$).
3) Sample a new value $\hat{y}$ from $q(y|\hat{x})$.
4) Set $\hat{x} \triangleq \hat{y}$ with probability p, where

$$p \triangleq \min\left\{\left(\frac{f(\hat{y})}{f(\hat{x})}\right)^{\alpha}, 1\right\} \tag{22}$$

5) Iterate 3–4 a "large" number of times.
6) Increase α according to some schedule.
7) Iterate 5–6 until convergence or until the available time is over.

The principle of simulating annealing is generic. It can be applied to *any* message-passing algorithm (e.g., sum-product algorithm, expectation maximization etc.), not only to the Metropolis-Hastings algorithm. The idea is to replace a local function f by a power f^α, where α increases in the course of the message-passing algorithm.

V. Conclusion

We presented particle methods as message passing on factor graphs. This viewpoint enables us to (i) combine various other families of signal-processing algorithms with particle methods in a disciplined manner; (ii) develop novel particle-based algorithms in a systematic fashion.

Acknowledgment

This project was supported in part by the Swiss National Science Foundation grant 200021-101955. The third author wishes to thank Ralf Koetter for helpful discussions on the message-passing interpretation of Gibbs sampling.

References

[1] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, Jan. 2004, pp. 28–41.
[2] H.-A. Loeliger, "Some remarks on factor graphs", *Proc. 3rd International Symposium on Turbo Codes and Related Topics*, 1–5 Sept., 2003, pp. 111–115.
[3] J. Dauwels, S. Korl, and H.-A. Loeliger, "Expectation maximization as message passing", *Proc. 2005 IEEE Int. Symp. Information Theory,* pp. 583–586.
[4] J. Dauwels, *On Graphical Models for Communications and Machine Learning: Algorithms, Bounds, and Analog Implementation,* PhD. Thesis at ETH Zurich, Diss. ETH No 16365, December 2005. Available from `www.dauwels.com/PhD.htm`.
[5] J. Dauwels, S. Korl, and H.-A. Loeliger, "Steepest descent on factor graphs," Proc. IEEE Information Theory Workshop, Rotorua, New Zealand, Aug. 28 – Sept. 1, 2005, pp. 42–46.
[6] H.-A. Loeliger, J. Hu, S. Korl, Qinghua Guo, and Li Ping, "Gaussian message passing on linear models: an update," *Proc. International Symposium on Turbo Codes and Related Topics,* Munich, Germany, April 3–7, 2006.
[7] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky "Non-parametric Belief Propagation," *CVPR,* June 2003.
[8] S. Korl, *A Factor Graph Approach to Signal Modelling, System Identification, and Filtering,* PhD. Thesis at ETH Zurich, Diss. ETH No 16170, July 2005.
[9] A. Doucet, J. F. G. de Freitas, and N. J. Gordon, eds., *Sequential Monte Carlo Methods in Practice.* New York: Springer-Verlag, 2001.
[10] A. Ihler, E. Sudderth, W. Freeman, and A. Willsky, "Efficient Multiscale Sampling from Products of Gaussian Mixtures," *NIPS,* Dec. 2003.
[11] C. Robert and G. Casella, *Monte Carlo Statistical Methods,* Springer Texts in Statistics, 2nd ed., 2004.