*In the Proc. of IEEE ISOC ITW2005 on Coding and Complexity; editor M.J. Dinneen; co-chairs U. Speidel and D. Taylor; pages 42-46*

# Steepest Descent as Message Passing

Justin Dauwels, Sascha Korl, and Hans-Andrea Loeliger
Dept. of Information Technology and Electrical Engineering,
ETH, CH-8092 Zürich, Switzerland
Email: {dauwels, korl, loeliger}@isi.ee.ethz.ch

***Abstract*— It is shown how steepest descent (or steepest ascent) may be viewed as a message passing algorithm with "local" message update rules. For example, the well-known backpropagation algorithm for the training of feed-forward neural networks may be viewed as message passing on a factor graph. The factor graph approach with its emphasis on "local" computations makes it easy to combine steepest descent with other message passing algorithms such as the sum/max-product algorithms, expectation maximization, Kalman filtering/smoothing, and particle filters. As an example, parameter estimation in a state space model is considered. For this example, it is shown how steepest descent can be used for the maximization step in expectation maximization.**

## I. Introduction

Suppose we want to find

$$\hat{\theta}_{\max} \triangleq \operatorname*{argmax}_{\theta} f(\theta), \tag{1}$$

where $\theta$ takes values in $\mathbb{R}^n$. The familiar steepest descent (or "steepest ascent" or "gradient descent/ascent") method tries to find $\hat{\theta}_{\max}$ as follows [1]: starting from some initial guess $\hat{\theta}^{(0)}$, compute

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda_k \, \nabla_\theta g(\theta)|_{\hat{\theta}^{(k)}} , \tag{2}$$

for $k = 1, 2, 3, \ldots$, where the parameter $\lambda_k$ (the "step size") is a positive real number that may be depend on $k$. An alternative update rule is

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda_k \, \nabla_\theta \log g(\theta)|_{\hat{\theta}^{(k)}} . \tag{3}$$

The update rule (2) or (3) is iterated until a fixed point is reached or until the available time is over.

In this paper, we will describe steepest descent as a message passing technique that operates on a factor graph. (See [2] for an introduction to factor graphs.) In particular, we will be interested in solving (1) in the case where $f(\theta)$ is a "marginal" of a real-valued function $f(x, \theta)$:

$$f(\theta) = \sum_x f(x, \theta), \tag{4}$$

where $\sum_x$ denotes either summation of integration over the whole range of $x$. We will assume $f(x, \theta) > 0$ for all $x$ and all $\theta$.

The described problem arises, for example, in the context of parameter estimation in state space models. In such a context, the variable $x$ is itself a vector and the function $f(x, \theta)$ has a nontrivial factor graph (cf., for example, Fig. 5). In such cases, the naive computation of (4) and/or (1) is often not feasible.

We will now assume that a factor graph for $f(x, \theta)$ is available. It may then be possible to compute $f(\theta)$ (4) and $\hat{\theta}_{\max}$ (1) by sum-product message passing and by max-product message passing, respectively [2]. Unfortunately, this approach is often unfeasible due to the following reasons:

- If the variable $x$ is continuous, the sum-product (integral-product) rule may lead to intractable integrals; in this case, we cannot compute (4).
- The max-product rule may lead to an intractable expression; in this case, we cannot compute (1).

One way to deal with the second problem (the maximization step) is to use steepest descent.

An alternative way to compute $\hat{\theta}_{\max}$ (1) (exactly or approximately) is expectation maximization (EM). (A message passing view of EM is developed in [4] [5].) However, the maximization step of EM is often intractable; we will show how steepest descent can be applied in such cases.

This paper is structured as follows. In Section 2, we describe steepest descent as message passing and its interaction with the sum-product algorithm. In Section 3, we demonstrate the maximization step of EM by means of steepest descent as message passing. Some concluding remarks are offered in Section 4.

## II. Sum-product algorithm and steepest descent

In earlier work [6], we briefly touched upon the subject of gradient descent in the context of the sum(mary)-product algorithm. Here, we present a more detailed exposition. As in [6], we start by considering the factor graph depicted in Fig. 1(a), which represents the global function $f(\theta) \triangleq f_A(\theta) f_B(\theta)$. The gradient $\nabla_\theta f(\theta)$ in update rule (2) is given by

$$\nabla_\theta f(\theta) \;=\; f_B(\theta) \nabla_\theta f_A(\theta) + f_A(\theta) \nabla_\theta f_B(\theta), \tag{5}$$

and similarly, the gradient $\nabla_\theta \log f(\theta)$ in update rule (3) equals

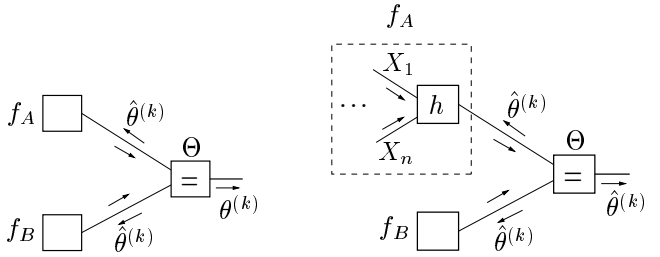
$$\nabla_\theta \log f(\theta) \;=\; \nabla_\theta \log f_A(\theta) + \nabla_\theta \log f_B(\theta). \tag{6}$$

Steepest descent according to rule (3) (and similarly (2)) may be viewed as follows:

1) The equality constraint node in Fig. 1(a) broadcasts the estimate $\hat{\theta}^{(k)}$. Node $f_A$ replies with the message $\nabla_\theta \log f_A(\theta)|_{\theta^{(k)}}$ and likewise node $f_B$.
2) A new estimate $\hat{\theta}^{(k+1)}$ is computed as

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda_k \Big( \nabla_\theta \log f_A(\theta) \Big|_{\hat{\theta}^{(k)}} + \nabla_\theta \log f_B(\theta) \Big|_{\hat{\theta}^{(k)}} \Big). \tag{7}$$

3) Iterate 1–2.




(a) $f_A$ and $f_B$ as closed boxes.

(b) Opening the box $f_A$.

Fig. 1. Factor graph of $f(\theta) = f_A(\theta) f_B(\theta)$.

In Fig. 1(a), the nodes $f_A$ and $f_B$ may be summaries of the subgraph "behind" them, as illustrated in Fig. 1(b): the function $f_A$ is a summary of the dashed box. This box contains a.o. the local node $h$, which is connected to the equality constraint node $\Theta$. The summary $f_A(\theta)$ is computed from the messages $\mu_{X_k \to h}$, arriving at the node $h$ from the left, according to the sum-product rule [2]:

$$f_A(\theta) \propto \sum_{x_1,\ldots,x_n} h(x_1,\ldots,x_n,\theta) \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to h}(x_\ell). \quad (8)$$

The above gradient method requires $\nabla_\theta f_A(\theta)$ and/or $\nabla_\theta \log f_A(\theta)$ (see (7)). In the following, we show how these expressions can be computed. We distinguish three cases:

1) $h$ is an equality constraint node
2) $h$ is differentiable
3) $h$ corresponds to a deterministic mapping.

### A. Equality constraint node


(a) Equality constraint node.

(b) Differentiable node function.

Fig. 2. Generic nodes.

If $h$ is an equality constraint node (see Fig. 2(a)), the required gradients are computed similarly as in (5) and (6):

$$\nabla_\theta f_A(\theta) = \sum_{\ell=1}^{n} \nabla_\theta \mu_{\theta_\ell \to \boxed{=}}(\theta) \prod_{m=1;m\neq\ell}^{n} \mu_{\theta_m \to \boxed{=}}(\theta), \quad (9)$$

and

$$\nabla_\theta \log f_A(\theta) = \sum_{\ell=1}^{n} \nabla_\theta \log \mu_{\Theta_\ell \to \boxed{=}}(\theta). \quad (10)$$

### B. Differentiable node function

Let $h(x_1,\ldots,x_n,\theta)$ be differentiable w.r.t. $\theta$. The gradient $\nabla_\theta f_A(\theta)$ can then be computed as follows:

$$\nabla_\theta f_A(\theta) \propto \sum_{x_1,\ldots,x_n} \nabla_\theta h(x_1,\ldots,x_n,\theta) \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to h}(x_\ell). \quad (11)$$

Note that in (11), we differentiated under the integral sign; we will always assume in this paper that this is allowed. The update rule (11) can be viewed as applying the sum-product rule to the node $\nabla_\theta h$, as illustrated in Fig. 2(b). The incoming messages are the standard sum-product summaries $\mu_{X_\ell \to h}$. In other words, if $h$ is differentiable, the differentiation operator does not propagate to the subgraph on the left of $h$; it is (only) applied to the local node function $h$. This is not the case if $h$ corresponds to a deterministic mapping, which is the subject of next subsection.

The gradient $\nabla_\theta \log f_A(\theta)$ equals

$$\nabla_\theta \log f_A(\theta) = \frac{\nabla_\theta f_A(\theta)}{f_A(\theta)}, \quad (12)$$

and is computed from (8) and (11). In order to evaluate $\nabla_\theta \log f_A(\theta)$, the sum-product rule is applied both to $h$ and $\nabla_\theta h$.

If the variables $X_\ell$ are discrete (and the alphabet is not "too large"), the expressions (11) and (12) can be evaluated in a straightforward manner. If on the other hand those variables (or a subset of them) are continuous, the integrals in (8) and (11) may be evaluated in several ways [6] (see also [7] for an illustration):

- In some cases, a closed-form expression of (8) or (11) exists.
- The integrals in (8) and (11) can be approximated based on canonical distributions as for example Gaussian distributions.
- The incoming messages $\mu_{X_\ell \to h}(x_\ell)$ may be a hard decision $\hat{x}_\ell$. The expressions (11) and (12) reduce to

$$\nabla_\theta f_A(\theta) \propto \nabla_\theta h(\hat{x}_1,\ldots,\hat{x}_n,\theta) \quad (13)$$

and

$$\nabla_\theta \log f_A(\theta) = \frac{\nabla_\theta h(\hat{x}_1,\ldots,\hat{x}_n,\theta)}{h(\hat{x}_1,\ldots,\hat{x}_n,\theta)}. \quad (14)$$

- The messages $\mu_{X_\ell \to h}(x_\ell)$ may be lists of samples (a.k.a. "particles") $\{x_\ell^{(i_\ell)}\}$ [6]. Consequently

$$\nabla_\theta f_A(\theta) \propto \sum_{i_1,\ldots,i_n} \nabla_\theta h\big(x_1^{(i_1)},\ldots,x_n^{(i_n)},\theta\big), \quad (15)$$

and

$$\nabla_\theta \log f_A(\theta) = \frac{\sum_{i_1,\ldots,i_n} \nabla_\theta h\big(x_1^{(i_1)},\ldots,x_n^{(i_n)},\theta\big)}{\sum_{i_1,\ldots,i_n} h\big(x_1^{(i_1)},\ldots,x_n^{(i_n)},\theta\big)}, \quad (16)$$

where the sums are taken over the lists of samples.

- An alternative method to solve the integrals in (8) and (11) is to quantize the variables $x_\ell$; the integrals become finite sums, which can be evaluated in a straightforward manner [6]. This method is however only applicable in low-dimensional systems.
- Combinations of the previous options are possible.

### C. Deterministic mapping

We consider the case where the local function $h$ corresponds to the deterministic mapping $y \stackrel{\triangle}{=} g(x_1, \ldots, x_n, \theta)$, i.e.,

$$h(x_1, \ldots, x_n, y, \theta) \stackrel{\triangle}{=} \delta\big(y - g(x_1, \ldots, x_n, \theta)\big). \tag{17}$$

We assume that $g$ is differentiable. Let $x \stackrel{\triangle}{=} (x_1, \ldots, x_n)$. The message $f_A(\theta)$ is computed as follows

$$f_A(\theta) \propto \sum_{x_1,\ldots,x_n} \delta\big(y - g(x,\theta)\big)\, \mu_{Y\to h}(y) \prod_{\ell=1}^{n} \mu_{X_\ell \to h}(x_\ell) \tag{18}$$

$$= \sum_{x} \mu_{Y\to h}\big(g(x,\theta)\big) \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to h}(x_\ell). \tag{19}$$

As a consequence

$$\nabla_\theta f_A(\theta) \propto \sum_x \nabla_\theta \big[\mu_{Y\to h}\big(g(x,\theta)\big)\big] \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to h}(x_\ell) \tag{20}$$

$$= \sum_x \nabla_\theta g(x,\theta)\ \nabla_y\, \mu_{Y\to h}(y)\big|_{y=g(x,\theta)} \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to h}(x_\ell). \tag{21}$$

Eq. (21) may be viewed as applying the sum-product rule to the node function $\nabla_\theta g$, as illustrated in Fig. 3. Note that the differentiation operator propagates to the left of $h$: besides the standard sum-product messages $\mu_{X_\ell\to h}(x_\ell)$, also the message $\nabla_y\, \mu_{Y\to h}(y)\big|_{g(x,\theta)}$ is required, which is the *gradient* of a sum-product message! The message $\nabla_y\, \mu_{Y\to h}(y)\big|_{g(x,\theta)}$ is computed by the same rules as $\nabla_\theta f_A(\theta)$ (cf. (9) (11) (21)). Similarly as in (11) and (12), the update rule (21) can be evaluated in several ways, depending on the datatype of the incoming messages. For example, if the incoming messages $\hat{x}_\ell$ and $\nabla_y\, \mu_{Y\to h}(y)\big|_{g(\hat{x},\theta)}$ are hard decisions, where $\hat{x}$ stands for $(\hat{x}_1, \ldots, \hat{x}_n)$, then

$$\nabla_\theta f_A(\theta) \quad \propto \quad \nabla_\theta g(\hat{x},\theta)\ \nabla_y\, \mu_{Y\to h}(y)\big|_{g(\hat{x},\theta)}\,. \tag{22}$$

This rule may be familiar to the reader who has some background in neural network theory: indeed, if one applies the rule (22) together with (7) (13) and (14) to a factor graph that represents a feed-forward neural network, one obtains the popular backpropagation algorithm [8].

### D. Summary

We have seen that

1) When steepest descent is combined with the sum-product algorithm, gradients of sum-product messages are required.

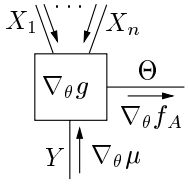



Fig. 3. Deterministic mapping $g$.

2) If the local node function $h$ is *differentiable*, the gradient of the outgoing message is computed by the sum-product rule applied to $\nabla_\theta h$, where the incoming messages are standard sum-product messages (see (11)). In other words, the differentiation operator does not propagate through the node $h$; it is only applied to the local node function $h$.
3) If the local node $h$ corresponds to a *deterministic mapping* $g$, the gradient of the outgoing message is computed by the sum-product rule applied to $\nabla_\theta g$ (see (21)). All incoming messages are standard sum-product messages, except for one, which is the *gradient* of an incoming sum-product message $\mu_Y$. In this case, the differentiation operator is applied to both the local node function and the incoming message $\mu_Y$; in other words, the differentiation operator *propagates* from node $h$ towards the node the message $\mu_Y$ has been sent from.
4) Differentiation also propagates through the equality constraint node (see (9) and (10)).
5) The three previous observations indicate that along an edge in the factor graph, the following messages may propagate
   - standard sum-product messages,
   - gradients of sum-product messages,
   - hard decisions $\hat{\theta}$,

   depending on
   - the location of the edges at which the steepest descent update rules are applied
   - the kind of nodes that are involved.
6) The sum-product messages and their gradients may be represented in various ways.

## III. Expectation maximization and steepest descent

In this section, we show how the maximization step in the EM algorithm can be performed by steepest descent. We start from the exposition in [5], where it is shown how the EM algorithm can be viewed as message passing on factor graphs (see also [4]). Consider the factorization

$$f(x,\theta) \stackrel{\triangle}{=} f_A(\theta) f_B(x,\theta), \tag{23}$$

which is represented by the factor graph of Fig. 4.

In this setup, EM amounts to iterative computation of the following messages [5]:

**Upward message $h(\theta)$:**

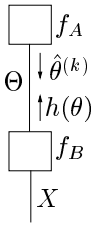



Fig. 4. Factor graph of (23).

$$h(\theta) = \frac{\sum_x f_B(x, \hat{\theta}^{(k)}) \log f_B(x, \theta)}{\sum_x f_B(x, \hat{\theta}^{(k)})} \tag{24}$$

**Downward message $\hat{\theta}^{(k)}$:**

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmax}}(\log f_A(\theta) + h(\theta)). \tag{25}$$

The computations (24) and (25) may be simplified when $f_A$ and $f_B$ have "nice" factorizations. Nevertheless, the maximization step (25) may still be intractable. We show by an example how gradient descent can be applied to solve this problem. Let

$$f_A(\theta) \triangleq f_{A_1}(\theta_1) f_{A_2}(\theta_1, \theta_2) \dots f_{A_n}(\theta_{n-1}, \theta_n), \tag{26}$$

and

$$\begin{aligned} f_B(x, \theta) \quad \triangleq \quad & f_{B_0}(x_0) f_{B_1}(x_0, x_1, y_1, \theta_1) f_{B_2}(x_1, x_2, y_2, \theta_2) \\ & \dots f_{B_n}(x_{n-1}, x_n, y_n, \theta_n), \end{aligned} \tag{27}$$

as illustrated in Fig. 5. As we pointed out before, the probability function $f(x, \theta)$ may represent a state-space model parameterized by the parameter $\theta$, whose prior model is determined by $f_A(\theta)$. The term $h(\theta)$ is given by [5]

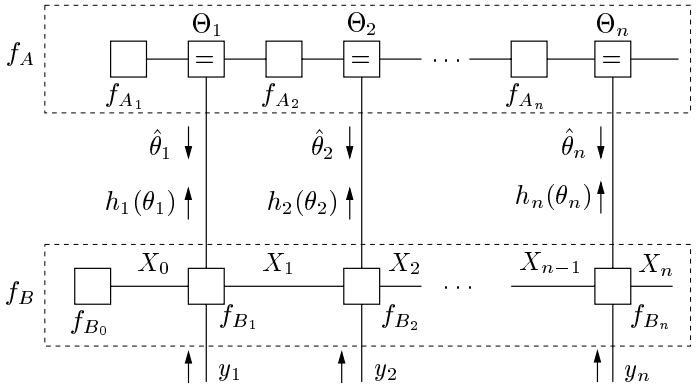



Fig. 5. Factor graph of (26) and (27).

$$h(\theta) \quad = \quad \sum_{\ell=1}^{n} h_\ell(\theta_\ell), \tag{28}$$

where

$$h_\ell(\theta_\ell) = \sum_{x_{\ell-1}, x_\ell} p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta}) \log f_{B_\ell}(x_{\ell-1}, x_\ell, y, \theta_\ell), \tag{29}$$

and $p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta})$ is the joint probability distribution of $X_{\ell-1}$ and $X_\ell$ (conditioned on $Y = y$ and $\Theta = \hat{\theta}$); the latter can be computed from the sum-product messages $\mu_{X_\ell \to f_{B_\ell}}(x_\ell)$ and $\mu_{X_{\ell-1} \to f_{B_\ell}}(x_{\ell-1})$ as follows

$$\begin{aligned} & p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta}) \\ & = \frac{f_{B_\ell}(x_{\ell-1}, x_k, y, \hat{\theta}_\ell) \mu_{X_\ell \to f_{B_\ell}}(x_\ell) \mu_{X_{\ell-1} \to f_{B_\ell}}(x_{\ell-1})}{\sum\limits_{x_{\ell-1}, x_\ell} f_{B_\ell}(x_{\ell-1}, x_\ell, y, \hat{\theta}_\ell) \mu_{X_\ell \to f_{B_\ell}}(x_\ell) \mu_{X_{\ell-1} \to f_{B_\ell}}(x_{\ell-1})}. \end{aligned} \tag{30}$$

The downward message $\hat{\theta}$ equals

$$\begin{aligned} & (\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_n)^T \\ & = \underset{\theta_1, \theta_2, \dots, \theta_n}{\operatorname{argmax}} \Big[ \log f_{A_1}(\theta_1) + \sum_{\ell=2}^{n} \log f_{A_\ell}(\theta_{\ell-1}, \theta_\ell) + \sum_{\ell=1}^{n} h_\ell(\theta_\ell) \Big]. \end{aligned} \tag{31}$$

The gradient $\nabla_\theta h(\theta) \triangleq \big(\nabla_{\theta_1} h(\theta), \dots, \nabla_{\theta_n} h(\theta)\big)^T$ required for steepest descent is computed as follows

$$\begin{aligned} & \nabla_{\theta_\ell} h_\ell(\theta_\ell) \\ & = \nabla_{\theta_\ell} \Big[ \sum_{x_{\ell-1}, x_\ell} p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta}) \log f_{B_\ell}(x_{\ell-1}, x_\ell, y, \theta_\ell) \Big], \end{aligned} \tag{32}$$

$$= \sum_{x_{\ell-1}, x_\ell} p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta}) \nabla_{\theta_\ell} \log f_{B_\ell}(x_{\ell-1}, x_\ell, y, \theta_\ell), \tag{33}$$

where $p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta})$ is given by (30).

Note that (30) (and hence also the rule (33)) involves standard sum-product messages. Those messages may again be represented in different ways, such as lists of particles, quantized messages, Gaussian distributions etc. [6].

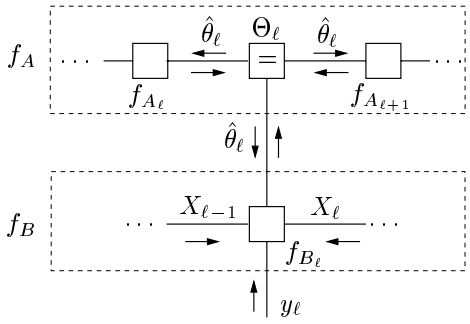



Fig. 6. Steepest descent as summary propagation.

Expectation maximization, in which the M-step is performed by steepest descent, may then be formulated as follows (see Fig. 6):

1) The equality constraint nodes $\Theta_\ell$ broadcast the estimates $\hat{\theta}_\ell^{(k)}$.
2) The nodes $f_{A_\ell}$ and $f_{A_{\ell+1}}$ reply with the messages $\nabla_{\theta_\ell} \log f_{A_\ell}|_{\hat{\theta}^{(k)}}$ and $\nabla_{\theta_\ell} \log f_{A_{\ell+1}}|_{\hat{\theta}^{(k)}}$ respectively.

3) A forward and backward sum(mary)-product sweep is performed in the box $f_B$.
4) The nodes $f_{B_\ell}$ reply with $\nabla_{\theta_\ell} h|_{\hat{\theta}^{(k)}}$, computed according to (33).
5) The new estimate $\hat{\theta}^{(k+1)}$ is computed:

$$\hat{\theta}_\ell^{(k+1)} = \hat{\theta}_\ell^{(k)} + \lambda_k \big( \nabla_{\theta_\ell} \log f_{A_\ell}|_{\hat{\theta}^{(k)}} + \nabla_{\theta_\ell} \log f_{A_{\ell+1}}\big|_{\hat{\theta}^{(k)}} + \nabla_{\theta_\ell} h|_{\hat{\theta}^{(k)}} \big). \quad (34)$$

6) Iterate 1–5.

Several update schedules are possible [2]. For example, in order to reduce the computational cost, one may prefer not to update the sum-product messages $\mu_{X_\ell \to f_{B_\ell}}(x_\ell)$ (cf. Step 3) at each iteration; the probability functions $p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta})$ (cf. Step 4) are then recomputed according to (30) using the *new* estimate $\hat{\theta}$, but the *old* messages $\mu_{X_\ell \to f_{B_\ell}}(x_\ell)$; this type of scheduling is the main idea behind [9]. Forward-only message passing amounts to recursive algorithms, known as "recursive EM" or "online EM" [10]–[13]. In [10] [11], recursive algorithms for fixed parameter estimation are derived based on EM in conjunction with steepest descent. It is common practice to extend the algorithms of [10] [11] to *time-varying* parameters by introducing some ad-hoc "forgetting" mechanism. We illustrated by the example (26)–(27), how parameters with non-trivial priors can be treated in a rigorous way (see also [4] [5] [12] [13]).

The example (26)–(27) can easily be extended to general functions $f_A$ and $f_B$. The gradient of the $h$-message leaving the generic node $g(z, \theta_m) \triangleq g(z_1, z_2, \ldots, z_n, \theta_m)$ (cf. Fig. 7) is given by

$$\nabla_{\theta_m} h(\theta_m) = \frac{\sum_z g(z, \hat{\theta}_m) \nabla_{\theta_m} \log g(z, \theta_m) \prod_{\ell=1}^{n} \mu_{Z_\ell \to \theta_m}(z_\ell)}{\sum_z g(z, \hat{\theta}_m) \prod_{\ell=1}^{n} \mu_{Z_\ell \to \theta_m}(z_\ell)}. \quad (35)$$

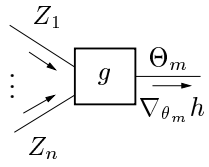



Fig. 7. Generic node $g$.

An illustration of the above procedure can be found in [14], where it is applied to the problem of code-aided carrier phase estimation.

## IV. Conclusion

We elaborated on previous work on steepest descent in the context of the sum(mary) product algorithm; we have shown in more detail how steepest descent can be viewed as message passing on factor graphs. In this setting, (1) steepest descent can easily be combined with other powerful algorithms such as Kalman filters/smoothers, the sum-product algorithm, expectation maximization and particle filters; (2) novel iterative signal processing algorithms can systematically be derived.

## V. Acknowledgments

This project was supported in part by the Swiss National Science Foundation grant 200021-101955.

## References

[1] D. P. Bertsekas, *Nonlinear programming,* Athena Scientific, Belmont, MA, 1995.
[2] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Proc. Mag.,* Jan. 2004, pp. 28–41.
[3] A. P. Dempster, N. M. Laird, and D. B. Rubin "Maximum likelihood from incomplete data via the EM algorithm,"*Journal of the Royal Statistical Society,* B 39, pp. 1–38, 1977.
[4] A. W. Eckford and S. Pasupathy, "Iterative multiuser detection with graphical modeling" *IEEE International Conference on Personal Wireless Communications,* Hyderabad, India, 2000.
[5] J. Dauwels, S. Korl, and H.-A. Loeliger, "Expectation maximization as message passing", *Proc. 2005 IEEE Int. Symp. Information Theory,* to appear.
[6] H.-A. Loeliger, "Some remarks on factor graphs", *Proc. 3rd International Symposium on Turbo Codes and Related Topics*, 1–5 Sept., 2003.
[7] J. Dauwels, and H.-A. Loeliger, "Phase estimation by message passing", *Proc. 2004 IEEE Int. Conf. on Communications,* June 20–24, Paris, France, pp. 523–527, 2004.
[8] C. M. Bishop, *Neural Networks for Pattern Recognition,* Oxford University Press, 1995.
[9] C. Herzet, V. Ramon and L. Vandendorpe, "Turbo synchronization: A Combined Sum-Product and Expectation-Maximization Algorithm Approach," *SPAWC'05 - IEEE Workshop on Signal Processing Advances in Wireless Communications,* June 5–8, 2005, New-York, USA.
[10] D. M. Titterington, "Recursive parameter estimation using incomplete data," R. Roy. Stat. Soc., vol. 46(B), pp. 256–267, 1984.
[11] E. Weinstein, M. Feder, A. V. Oppenheim, "Sequential algorithms for parameter estimation based on the Kullback-Leibler information measure," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 38, Sept. 1990, pp. 1652-1654.
[12] H. Zamiri-Jafarian and S. Pasupathy, "EM-based recursive estimation of channel parameters," *IEEE Transactions on Communications,* vol. 47, Sept. 1999, pp. 1297–1302.
[13] L. Frenkel and M. Feder, "Recursive estimate-maximize (EM) algorithms for time varying parameters with applications to multiple target tracking," *IEEE International Conference on Acoustics, Speech, and Signal Processing,* vol. 3, 9–12 May, 1995, pp. 2068–2071.
[14] J. Dauwels, S. Korl, and H.-A. Loeliger, "Expectation maximization for phase estimation," *Proc. Eighth International Symposium on Communication Theory and Applications (ISCTA'05),* to appear.