Comments for "Choosing a Variable to Clamp: Approximate Inference Using Conditioned Belief Propagation"

Frederik Eaton

Dec 28, 2011

Contents

1 Errata **1**

2 Concerning the conditioning idea **2**

3 General comments concerning automatic differentiation **2**

3.1 Isomorphism of forward and reverse mode automatic differentiation on BP . . . 2

3.2 Future work . . . 7

References **8**

Abstract

This document will mainly be of interest to those who are re-implementing or extending "Choosing a Variable to Clamp" [1]. Most of the text is concerned with demonstrating a fairly straightforward isomorphism between forward and reverse-mode automatic differentiation applied to the belief propagation algorithm, in section 3. This shows that "Back-Belief Propagation" of [1] is performing similar updates to the "Linear Response" algorithm [3].

1 Errata

We are not aware of any mistakes in the algorithm or the main conclusions of "Choosing a Variable to Clamp" [1], but there is a minor typo as well as an unintentional misrepresentation of the performance of a competing algorithm; these are described below.

1. Third page, lower-right: "Algorithm (BPP)" should read "Algorithm (BBP)"
2. Figure 3: Our GBP performance measurements were based on the GBP implementation in libDAI, which uses the inner loop of HAK, rather than the parent-child updates of Yedidia et al 2001. In many cases the parent-child updates lead to more accurate marginals, and are also faster. Thus, the points labelled "GBP" in this figure should not be regarded as a fair evaluation of Yedidia's original algorithm.

2 Concerning the conditioning idea

The chapter in Eaton's PhD thesis which presents the work of "Choosing a Variable to Clamp" (CAVC) [1] includes some further comments, in the discussion section, on the practicality of the recursive conditioning idea and on future directions in which it could be developed.

3 General comments concerning automatic differentiation

In "Implicit Differentiation by Perturbation", NIPS 2010 [2], Justin Domke shows that it is possible to use the standard numerical perturbation method ($f'(x) \approx \frac{f(x+h)-f(x)}{h}$) to get a full set of adjoints at the potentials of a graph, given adjoints at the marginals. This technique requires only two runs of inference and can be applied to any approximate inference algorithm, not just BP. It is based on a trick which leverages the symmetry of the Jacobian of the function which relates potentials and marginals. This Jacobian is symmetric even for the approximate marginals of belief propagation and mean field, as shown by Welling and Teh, "Linear Response Algorithms for Approximate Inference in Graphical Models" [3]. Domke demonstrates empirically that although the accuracy of his perturbation method is often similar to that of BBP, his method is slightly faster.

Domke also relates that BBP is much faster than generic automatic differentiation libraries such as ADOLC in his experiments, perhaps by a factor of ten, while using much less memory (such libraries must keep a record of every message update). These observations are based on his TRW variant of BBP (he calls it "BTRW"), but should hold in general. Domke uses BTRW, rather than his earlier perturbation method, in his paper "Parameter Learning with Truncated Message-Passing" (CVPR 2011) [4], since it is able to report a useful gradient without full convergence of the messages.

3.1 Isomorphism of forward and reverse mode automatic differentiation on BP

Here we elaborate on one of the implications of Domke's insight, which we first describe in more detail. The Jacobian in question is

$$\frac{\partial b_i(x_i)}{\partial \log \psi_j(x'_j)} = \frac{\partial b_j(x'_j)}{\partial \log \psi_i(x_i)}$$

and obeys the indicated symmetry. Below, we will write θ for $\log \psi$ for brevity.

Now, if we want to calculate the change in the beliefs of a model, db, given a small perturbation in the log-factors dθ, we should use matrix multiplication as follows:[1]

$$\mathrm{d}b_i(x_i) = \sum_{j,x'_j} \frac{\partial b_i(x_i)}{\partial \theta_j(x'_j)} \mathrm{d}\theta_j(x'_j) \tag{1}$$

We will usually omit the variable arguments x, so that this equation can be rewritten more concisely as

$$\mathrm{d}b_i = \sum_{j,\mathcal{X}_j} \frac{\partial b_i}{\partial \theta_j} \mathrm{d}\theta_j \tag{2}$$

(we have indicated summation over the omitted argument x'_j by its domain $\mathcal{X}_j$).

If, on the other hand, we have an objective function $V(b)$ and want to determine the gradient of V with respect to θ

$$\frac{\mathrm{d}V}{\mathrm{d}\theta_j} \equiv \not{d}\theta$$

given the gradient $\not{d}b$ of V with respect to b, then we should multiply the input vector by the transpose of the Jacobian matrix:

$$\not{d}\theta_j = \sum_{i,\mathcal{X}_i} \frac{\partial b_i}{\partial \theta_j} \not{d}b_i \tag{3}$$

Since the Jacobian matrix is symmetrical, $\frac{\partial b_i}{\partial \theta_j} = \frac{\partial b_j}{\partial \theta_i}$, it follows that we can accomplish the ("reverse-mode") calculation of equation 3 using any algorithm that calculates ("forward-mode") equation 1. All we have to do is give the gradient inputs $\not{d}b$ as the differential inputs $\mathrm{d}\theta$, and then interpret the differential outputs $\mathrm{d}b$ as gradients $\not{d}\theta$. Domke [2] uses numerical differentiation to do this:

$$\mathrm{d}b \approx \frac{f(\theta + h\mathrm{d}\theta) - f(\theta)}{h} \tag{4}$$

or equivalently

$$\not{d}\theta \approx \frac{f(\theta + h\not{d}b) - f(\theta)}{h} \tag{5}$$

where f is a function that computes BP marginals given potentials and h is a very small number.

In Welling and Teh's paper [3], an algorithm is given for calculating $\frac{\partial b_i}{\partial \theta_j}$ for every (i, j). This algorithm is similar to n parallel runs of "forward-mode automatic differentiation" (FAD). A single run of FAD will calculate $\mathrm{d}b_i = \sum_{j,\mathcal{X}_j} \frac{\partial b_i}{\partial \theta_j} \mathrm{d}\theta_j$ for every i but only a single vector $\mathrm{d}\theta$, with computational cost proportional to one run of BP. The algorithm presented by Welling and Teh has computational cost proportional to n runs of BP (where n is the number of variables in the model), assuming a constant variable domain size.

[1] If the reader is uncomfortable with differentials like $\mathrm{d}b$, he can think of them as representing derivatives along a curve parameterised by an independent variable, say u. Then θ is a function $\theta(u)$ of u, and one should read $\mathrm{d}\theta$ as the rate of change $\frac{\mathrm{d}\theta}{\mathrm{d}u}$. Then $b = b(\theta(u))$ is also a function of u, and $\mathrm{d}b$ is the corresponding rate $\frac{\mathrm{d}b}{\mathrm{d}u}$ which can be expanded via the chain rule to yield equation 1.

In CAVC we introduce the idea of performing “reverse-mode automatic differentiation”, or RAD, which is the same as back-propagation, on BP to calculate $\not{d}\theta_j = \sum_{i,\mathcal{X}_i} \frac{\partial b_i}{\partial \theta_j} \not{d} b_i$. We called the resulting algorithm BBP. However, in light of Domke’s insight mentioned above, we see that BBP is only calculating the same quantities as FAD, which is in turn a straightforward specialisation of Welling and Teh’s algorithm. The question naturally arises as to whether BBP, i.e. RAD on BP, is the in fact same algorithm as FAD on BP; or whether they are calculating the same quantities in different ways. The latter scenario would be surprising, and it does not prove to be the case.

We give a proof of the isomorphism between RAD and FAD on BP below. It may well be that there is a simpler way to show that these two algorithms are equivalent, but it is not immediately clear that the equivalence is a trivial one. In the first case, FAD propagates the BP messages forward through time, while RAD propagates them backwards. Also, in order to obtain the equivalence it was necessary to sum the RAD messages over time, to switch messages with their duals, and to multiply the FAD messages by the node beliefs. Nevertheless, we suspect that the presentation below may benefit from some further simplifying insight. At the moment, we only give a straightforward derivation of both algorithms, and put them into a form where it is easy to see that they are doing the same thing.

The following uses the same notation as CAVC [1], which is described there in more detail.

First we review the BP messages, in dual form:

$$\bar{n}_{i\alpha} \leftarrow \psi_i \prod_{\beta\sim i\setminus\alpha} m_{\beta i} \qquad n_{i\alpha} = \frac{1}{Z_{n_{i\alpha}}} \bar{n}_{i\alpha} \tag{6}$$

$$\bar{m}_{\alpha i} \leftarrow \sum_{\mathcal{X}_{\alpha\setminus i}} m^F_{\alpha i} \qquad m_{\alpha i} = \frac{1}{Z_{m_{\alpha i}}} \bar{m}_{\alpha i} \tag{7}$$

$$m^F_{\alpha i} = \psi_\alpha \prod_{j\sim\alpha\setminus i} n_{j\alpha} \tag{8}$$

where we have omitted variable arguments as in equation 2, and where the intermediate quantity $m^F_{\alpha i}(x_\alpha)$ has been introduced to simplify the later derivations.

Although the BBP algorithm presented in CAVC is written in a form which allows factors with zero entries, the messages are greatly simplified if we assume strict positivity and manipulate derivatives of logarithms of the usual quantities. Note that

$$\mathrm{d}\log q = \frac{1}{q}\mathrm{d}q \tag{9}$$

We also derive the differential normalisation rule (see CAVC equation 13) for logarithms: given

$$q(x) = \frac{\bar{q}(x)}{\sum_{x'} \bar{q}(x') \equiv Z_q} \tag{10}$$

we have the relation

$$\mathrm{d}q(x) = \frac{1}{Z_q}(\mathrm{d}\bar{q}(x) - q(x)\sum_{x'} \mathrm{d}\bar{q}(x')) \tag{11}$$

The "logarithmic" form of this normalisation is

$$\mathrm{d}\log q(x) = \mathrm{d}\log \bar{q}(x) - \sum_{x'} q(x')\mathrm{d}\log \bar{q}(x') \tag{12}$$

Applying the standard rules of differentiation to the above messages, and writing $\mu \equiv \log m$ and $\nu \equiv \log n$ (and e.g. $\bar{\mu} \equiv \log \bar{m}$), we find that

$$\mathrm{d}\bar{\nu}_{i\alpha} \leftarrow \sum_{\beta\sim i\backslash\alpha} \mathrm{d}\mu_{\beta i} + \mathrm{d}\theta_i \tag{13}$$

$$\mathrm{d}\bar{\mu}^F_{\alpha i} \leftarrow \sum_{j\sim\alpha\backslash i} \mathrm{d}\nu_{j\alpha} + \mathrm{d}\theta_\alpha \tag{14}$$

and from $\mathrm{d}\bar{m}_{\alpha i} = \sum_{\mathcal{X}_{\alpha\backslash i}} \mathrm{d}\bar{m}^F_{\alpha i}$ we have

$$\mathrm{d}\bar{\mu}_{\alpha i} = \sum_{\mathcal{X}_{\alpha\backslash i}} \frac{\bar{m}^F_{\alpha i}}{\bar{m}_{\alpha i}} \mathrm{d}\bar{\mu}^F_{\alpha i} \tag{15}$$

Noting that $\bar{m}^F_{\alpha i} = \bar{b}_\alpha / n_{i\alpha}$ and $\bar{m}_{\alpha i} n_{i\alpha} = \sum_{\mathcal{X}_{\alpha\backslash i}} \bar{b}_\alpha$ we have

$$\frac{\bar{m}^F_{\alpha i}}{\bar{m}_{\alpha i}} = \frac{\bar{b}_\alpha}{\sum_{\mathcal{X}_{\alpha\backslash i}} \bar{b}_\alpha} = \frac{b_\alpha}{\sum_{\mathcal{X}_{\alpha\backslash i}} b_\alpha} = \frac{b_\alpha}{b_i} \tag{16}$$

Combining this with 14 and 15 gives

$$\mathrm{d}\bar{\mu}_{\alpha i} \leftarrow \sum_{\mathcal{X}_{\alpha\backslash i}} \frac{b_\alpha}{b_i} \left(\sum_{j\sim\alpha\backslash i} \mathrm{d}\nu_{j\alpha} + \mathrm{d}\theta_\alpha \right) \tag{17}$$

Equations 17 and 13 give the FAD BP message updates. The final differentials are straightforward from the beliefs:

$$\mathrm{d}\log \bar{b}_i = \mathrm{d}\theta_i + \sum_{\alpha\sim i} \mathrm{d}\mu_{\alpha i} \tag{18}$$

$$\mathrm{d}\log \bar{b}_\alpha = \mathrm{d}\theta_\alpha + \sum_{i\sim\alpha} \mathrm{d}\nu_{i\alpha} \tag{19}$$

Now for the RAD updates it is necessary to introduce the idea of summing the gradients over time, to get an "integrated gradient" quantity which we shall write with $\not{d}^\Sigma$ and define as:

$$\not{d}^\Sigma q \equiv \sum_t \not{d} q^t \tag{20}$$

Again we use logarithmic messages, but this time note that in contrast to equation 9 we have

$$\not{d} \log q = q \not{d} q \tag{21}$$

We also derive reverse-mode normalisation rules. The following appears in CAVC:

$$\not{d}\bar{q}(x) = \frac{1}{Z_q}(\not{d}q(x) - \sum_{x'} q(x')\not{d}q(x')) \tag{22}$$

but here we only use its logarithmic version

$$\not d \log \bar q(x) = \not d \log q(x) - \sum_{x'} q(x') \not d \log q(x') \tag{23}$$

Note the close similarity of equation 23 to equation 12 even though equations 22 and 11 are somewhat different.

Assuming that the BP run terminates at time T, the dependencies on the m messages result in gradient updates:

$$\not d m_{\alpha i}^{t-1} = \sum_{\beta\sim i\setminus\alpha} \frac{\bar n_{i\beta}^{t}}{m_{\alpha i}^{t-1}} \not d \bar n_{i\beta}^{t} + \delta_t^T \frac{\bar b_i}{m_{\alpha i}^{t-1}} \not d \bar b_i \tag{24}$$

or equivalently

$$\not d \mu_{\alpha i}^{t-1} = \sum_{\beta\sim i\setminus\alpha} \not d \bar\nu_{i\beta}^{t} + \delta_t^T \not d \log \bar b_i \tag{25}$$

Summing over t gives

$$\not d^{\Sigma} \mu_{\alpha i} \leftarrow \sum_{\beta\sim i\setminus\alpha} \not d^{\Sigma} \bar\nu_{i\beta} + \not d \log \bar b_i \tag{26}$$

Similarly for dependencies on n we get

$$\not d^{\Sigma} \nu_{i\alpha} \leftarrow \sum_{\mathcal{X}_{\alpha\setminus i}} \sum_{j\sim\alpha\setminus i} \not d^{\Sigma} \bar\mu_{\alpha j}^{F} + \not d \log \bar b_\alpha \tag{27}$$

where

$$\not d^{\Sigma} \bar\mu_{\alpha i}^{F} = \frac{\bar m_{\alpha i}^{F}}{\bar m_{\alpha i}} \not d^{\Sigma} \bar\mu_{\alpha i} = \frac{b_\alpha}{b_i} \not d^{\Sigma} \bar\mu_{\alpha i} \tag{28}$$

Equation 29 becomes

$$\not d^{\Sigma} \nu_{i\alpha} \leftarrow \sum_{\mathcal{X}_{\alpha\setminus i}} \left(\sum_{j\sim\alpha\setminus i} \frac{b_\alpha}{b_j} \not d^{\Sigma} \bar\mu_{\alpha j} + \not d \log \bar b_\alpha \right) \tag{29}$$

Finally,

$$\not d \theta_i = \not d \log b_i + \sum_{\alpha\sim i} \not d^{\Sigma} \bar\nu_{i\alpha} \tag{30}$$

$$\not d \theta_\alpha = \not d \log b_\alpha + \sum_{i\sim\alpha} \not d^{\Sigma} \bar\mu_{\alpha i} \tag{31}$$

Writing the two pairs of message updates one above the other

$\not d^{\Sigma} \mu_{\alpha i} \leftarrow \sum_{\beta\sim i\alpha} \not d^{\Sigma} \bar\nu_{i\beta} + \not d \log \bar b_i$	$\not d^{\Sigma} \nu_{i\alpha} \leftarrow \sum_{\mathcal{X}_{\alpha i}} \left(\sum_{j\sim\alpha i} \frac{b_\alpha}{b_j} \not d^{\Sigma} \bar\mu_{\alpha j} + \not d \log \bar b_\alpha \right)$
$\updownarrow$	$\updownarrow$
$\mathrm{d}\bar\nu_{i\alpha} \leftarrow \sum_{\beta\sim i\alpha} \mathrm{d}\mu_{\beta i} + \mathrm{d}\theta_i$	$\mathrm{d}\bar\mu_{\alpha i} \leftarrow \sum_{\mathcal{X}_{\alpha i}} \frac{b_\alpha}{b_i} \left(\sum_{j\sim\alpha i} \mathrm{d}\nu_{j\alpha} + \mathrm{d}\theta_\alpha \right)$

and multiplying the bottom two updates by b_i, we see that the correspondence $\not{d}^{\Sigma}\mu_{\alpha i} \leftrightarrow b_i \mathrm{d}\nu_{i\alpha}$ and $\not{d}^{\Sigma}\nu_{i\alpha} \leftrightarrow b_i \mathrm{d}\mu_{\alpha i}$ relates the two pairs of updates. Although the normalisations are different (for instance $\not{d}^{\Sigma}\mu_{\alpha i}$ is normalised to have a zero inner product with $m_{\alpha i}$, while $b_i \mathrm{d}\nu_{i\alpha}$ is normalised to have a zero inner product with $\frac{1}{m_{\alpha i}}$), normalisation involves adding a constant which is forgotten after each update and can therefore be seen as arbitrary (as in Welling and Teh [3]). Thus, applying FAD and RAD to BP yields isomorphic algorithms. We leave the relationship between equations 30 and 18, and 31 and 19, as an exercise to the reader.

Which form of updates should you use? We haven't done any experiments to, for instance, compare the speed or numerical stability of the two representations. The FAD algorithm of Welling and Teh requires some modification if you just want to calculate a single perturbation. And their algorithm as they state it uses division by messages, so additional simple modifications must be made to support factors with zero entries. Our BBP algorithm, on the other hand, is expressed in CAVC in terms of "additive" (rather than "destructive") updates. In other words, in the original formulation of BBP, messages converge to zero over time and the results of each update are summed into "accumulators". The "integrated" form of the message quantities, introduced here in equation 26 using the $\not{d}^{\Sigma}$ notation of equation 20, allowed us to rewrite BBP using "destructive" updates, so that messages converge to fixed values and the output is a function of these values. Again, we haven't done any experiments comparing the two update styles, but the destructive updates seem potentially more stable, while being explicitly flexible as to the update order, allowing for a clearer presentation and implementation than those of CAVC. As with Welling and Teh's updates, if one wishes to support zero entries, the updates presented in this note are not suitable, and one must rewrite them in a form which is similar to that in CAVC - avoiding logarithms and divisions.

3.2 Future work

There are a few open questions surrounding these ideas. Is there an easier way to show the equivalence of RAD and FAD on BP? Presumably RAD and FAD are also equivalent on mean field; what about other message passing algorithms such as Joris Mooij's LCBP [5] (for which we are not aware of an energy functional)? Can we derive some structural property which a message passing algorithm should possess in order to satisfy this equivalence?

Can we extend the notion of an equivalence between forward and reverse modes of differentiation, to that of an equivalence between forward and reverse versions of an algorithm itself? This may lead to an interesting connection between message-passing algorithms and sampling algorithms like MCMC: We might think that running a convergent message-passing algorithm backwards (if we can come up with a sensible way to implement such a concept) could lead to an unstable, chaotic system, perhaps similar to Herding (e.g. Welling 2009 [6]). This in turn suggests that it may be possible to combine a message-passing algorithm and a sampling algorithm in such a way that the result is invariant under time-reversal (i.e. if the pseudo-"sampling" component is obtained by reversing the "message-passing" component).

Most of the citations of CAVC are by people who are using the BBP algorithm. So far, all of these citations are for applications where graph potentials are to be learned given a sample of data. Such learning-oriented applications of approximate inference are typical of mainstream machine learning. The primary contribution of CAVC, in contrast, was intended to be an algorithm in the "pure approximate inference" domain, where a model's potentials have already been learned (or are specified by rules) and marginals must be estimated without validation by data. In retrospect it was not surprising that BBP, rather than the recursive conditioning approximation technique to which it was applied as a variable choice heuristic, has the greater number of users. We are still periodically interested in recursive conditioning, and in particular the problem of making it practical on models with weak coupling. At the same time, we would like to see more serious applications of BBP, as well as LR and other automatic differentiation techniques, in the pure approximate inference setting. The "Fractional Belief Propagation" paper of Tom Heskes [7] uses LR in a more natural way than CAVC uses BBP, yet neither application is quite satisfying, and one imagines that more interesting techniques are waiting to be discovered. By working to clarify the relationship between (reverse-mode) BBP and (forward-mode) LR, we hope to have removed a potential stumbling block to further research in this area.

References

[1] F. Eaton and Z. Ghahramani. Choosing a variable to clamp: approximate inference using conditioned belief propagation. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 145–152, 2009.

[2] Justin Domke. Implicit differentiation by perturbation. In *Advances in Neural Information Processing Systems 23*, pages 523–531. 2010.

[3] M. Welling and Y.W. Teh. Linear response algorithms for approximate inference in graphical models. *Neural computation*, 16(1):197–221, 2004.

[4] J. Domke. Parameter learning with truncated message-passing. pages 2937–2943, 2011.

[5] J.M. Mooij, B. Wemmenhove, HJ Kappen, and T. Rizzo. Loop corrected belief propagation. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[6] M. Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128. ACM, 2009.

[7] W. Wiegerinck and T. Heskes. Fractional belief propagation. In *Advances in Neural Information Processing Systems 15*, page 455. MIT Press, 2003.