Local Training and Belief Propagation

Charles Sutton and Tom Minka

Microsoft Research Ltd.

MSR-TR-2006-121

10 August 2006

Abstract

Because maximum-likelihood training is intractable for general factor graphs, an appealing alternative is *local training*, which approximates the likelihood gradient without performing global propagation on the graph. We discuss two new local training methods: shared-unary piecewise, in which unary factors are shared among every higher-way factor that they neighbor, and the one-step cutout method, which computes exact marginals on overlapping subgraphs. Comparing them to naive piecewise training, we show that just as piecewise training corresponds to using the Bethe pseudomarginals after zero BP iterations, shared-unary piecewise corresponds to the pseudomarginals after one parallel iteration, and the one-step cutout method corresponds to the beliefs after two iterations. We show in simulations that this point of view illuminates the errors made by shared-unary piecewise.

1 Introduction

One approach for approximate parameter estimation in loopy factor graphs is *local training*, as in pseudolikelihood [2] and piecewise training [6]. Another intuitive approach is to compute approximate marginal distributions using a global propagation algorithm like BP, and simply substitute the resulting beliefs into the exact gradient. These approaches are actually closely related, because stopping after the first few iterations of BP will yield messages that depend only on local information in the graph. In this note, we introduce two new local training algorithms, and show that they can be viewed either as performing local training, or as computing approximate marginals using messages from early iterations of BP.

The new algorithms are *shared-unary piecewise*, which is standard piecewise with unary factors shared among the pieces, and *one-step cutout*, which computes exact marginals in overlapping subgraphs of the original graph. In order to see how these algorithms arise from BP, we need to make a distinction that is not always clear in the literature—the distinction between beliefs and pseudomarginals. By the *belief* of a node i, we mean its normalized product of messages, just as one ordinarily computes beliefs after running BP. By the *pseudomarginal*, on the other hand, we mean the derivative of the free energy with respect to the unary parameters of the model. Although when BP has converged the beliefs and pseudomarginals are equal, they are in general distinct, and we can obtain insight by examining algorithms from both of these viewpoints.

Each of the local training algorithms can then be interpreted in three ways: as performing maximum-likelihood training on a modified graph, which we call the *neighborhood graph view*; as maximizing the Bethe energy at a particular message setting, which we call the *pseudomarginal view*; or as approximating the ML gradient using beliefs at a particular message setting of BP, which we call the *belief view*. These viewpoints are listed in Table 2, which summarizes much of this note. In particular, just as standard piecewise corresponds to zero iterations of BP, shared-unary piecewise corresponds to one

iteration, and one-step cutout under certain conditions corresponds to two iterations. (For reference, the messages after zero, one, and two BP iterations are given in Table 1.)

Thus, the contributions of this note are as follows. First, we explain the local training, pseudomarginal, and belief perspectives of these algorithms (Section 3). Then we present shared-unary piecewise (Sections 4 and 5) and one-step cutout (Section 6), explaining them both from the local training and BP perspectives. By taking the BP viewpoint, we show how the pseudomarginals of shared unary can be viewed as an approximation to the one-step cutout beliefs (Section 7). This viewpoint provides an explanation of the approximation error made by shared-unary piecewise in simulated data (Section 8).

2 Background

Consider a CRF $p(\mathbf{y}|\mathbf{x})$ with parameters θ and terms $t_a(y_a, \mathbf{x}, \theta)$, where a denotes a factor, and y_a an assignment to the variables in a. We assume that p is *weakly canonical* form, by which we mean that every variable i has a unary term $t_i(y_i, \mathbf{x}, \theta)$, and for every non-unary factor a and variable $i \in a$, the sum $\sum_{y_a \setminus y_i} t_a(y_a, \mathbf{x}, \theta)$ is uniform over y_i. For fixed $\mathbf{x}$, this transformation is always possible, and it means intuitively that none of the unary information is hidden within higher-way factors. Note that this is a weaker condition than the canonical form used in the Hammersley-Clifford theorem [1], in that we allow k-way factors ($k > 2$) to contain $(k-1)$-way information; for example, a three-way factor may contain two-way information. We assume that the unary factors are parameterized as

$$t_i(y_i, \mathbf{x}, \theta) = \exp\{\theta_i(y_i)\}. \tag{1}$$

BP approximates the CRF partition function as (using the dual Bethe energy introduced in [3]):

$$Z_{\text{BP}}(\mathbf{x}, \theta) = \max_{\tilde{t}_a(y_a)} \prod_a \left(\sum_{y_a} \frac{t_a(y_a, \mathbf{x}, \theta)}{\tilde{t}_a(y_a)} q(y_a) \right) \prod_i \left(\sum_{y_i} q(y_i) \right)^{1-d_i}, \tag{2}$$

where we define the unnormalized beliefs as

$$q(\mathbf{y}) = \prod_a \tilde{t}_a(y_a) = \prod_a \prod_i m_{ai}(y_i), \tag{3}$$

with $q(y_a) = \sum_{\mathbf{y} \setminus y_a} q(\mathbf{y})$ and $q(y_i) = \sum_{\mathbf{y} \setminus y_i} q(\mathbf{y})$. We define the normalized beliefs as $b_a(y_a) \propto q(y_a)$.

If we set the $\tilde{t}_a$ to 1, as if we had performed zero iterations of BP, then we get the piecewise approximation [6] (up to a constant):

$$Z_{\text{PW}}(\mathbf{x}, \theta) = \prod_a \left(\sum_{y_a} t_a(y_a, \mathbf{x}, \theta) \right) \prod_i \left(\sum_{y_i} 1 \right)^{1-d_i}. \tag{4}$$

To incorporate unary factors, one way is to train the unary factors separately, like any other piece, but sometimes this leads to a bad approximation. In previous work on piecewise training [6], the models contained no unary factors, using instead fully-parameterized pairwise factors. This sidestepped the question of how to deal with unary factors. In Sections 4 and 6, we discuss alternative approximations that approximate $\log Z$ better.

For more information on CRFs and approximate likelihoods, see [4, 7].

3 Pseudomarginals, Beliefs, and Neighborhood Graphs

In this section, we explain the subgraph and BP viewpoints on local training algorithms. First, many local training algorithms are straightforwardly viewed as performing exact inference on a transformed graph that cuts the global dependencies in the model. For example, standard piecewise performs

maximum-likelihood training in a *node-split graph* in which variables are duplicated so that each factor is in its own connected component [5]. We refer to this viewpoint as the *neighborhood graph view* of a training algorithm.

Second, many local training algorithms can be interpreted as approximating $\log Z$ by the Bethe energy $\log Z_{\text{BP}}$ at a particular message setting. We call this the *pseudomarginal view*, because under this view, the estimated parameters are chosen to match the pseudomarginals to the empirical marginals.

Finally, any approximate inference algorithm can be used to perform approximate ML training, by substituting the approximate beliefs for the exact marginals in the ML gradient. We call this the *belief view* of an approximate training algorithm. For example, this is the standard way of implementing approximate training using BP. So any method for computing approximate beliefs yields an approximate gradient. Interestingly, not all such approximations to the gradient can themselves be obtained as the exact gradient of any single approximate objective function. An example of an approximate gradient that has no objective function is the one-step cutout method (Section 6). Recently, training methods that have a pseudomarginal interpretation have received much attention, but it is not clear if training methods that have a pseudomarginal interpretation should be preferred over ones that do not.

The pseudomarginal and belief viewpoints are distinct. To explain this, we need to make a distinction that is not always clear in the literature, between beliefs and pseudomarginals. By the *belief* of a node i, we mean its normalized product of messages, which is proportial to $q(y_i)$. By *pseudomarginal*, on the other hand, we mean the derivative of $\log \tilde{Z}$ with respect to θ_i. These quantities are distinct. For example, in standard piecewise, the pseudomarginal $\partial \log Z_{\text{PW}}/\partial\theta_i(y_i)$ equals $p_i(y_i)$, but the belief is proportional to $q(y_i) = \sum_{\mathbf{y}\setminus y_i} q(y_i) = 1$.

This point may be confusing for several reasons. First, when the messages $\tilde{t}$ are a fixed point of BP, then the pseudomarginal always equals the belief. But this does not hold before convergence, and we shall be concerned in this note with intermediate message settings, before BP has converged. A second potential confusion arises because we define Z_{BP} using the dual Bethe energy [3] rather than the primal. In the primal Bethe energy [9], the pseudomarginal equals the belief at all message settings, but this is not true of the dual energy. We use the dual energy rather than the primal not only because it helps in interpreting local training algorithms, but also because at intermediate message settings it tends to be a better approximation to $\log Z$.

Finally, when calculating pseudomarginals $\partial \log \tilde{Z}/\partial\theta_i$, we must recognize that the message setting is often itself a function of θ. For example, suppose we stop BP after one iteration, that is, we take $\tilde{Z}(\theta) = Z_{\text{BP}}(\theta, \tilde{t}^{(1)}(\theta))$, where $\tilde{t}^{(1)}$ are the messages after one BP iteration. Then, because the message setting is clearly a function of θ, we need to take $\partial\tilde{t}^{(1)}/\partial\theta_i$ into account when computing the pseudomarginals of $\tilde{Z}$.

4 Shared-Unary Piecewise

One idea for improving the piecewise estimate of the unary gradient is to duplicate the unary factors over every non-unary piece that they neighbor. This yields *shared-unary piecewise*. Recall that the standard piecewise Z_{PW} arises from Z_{BP} when all $\tilde{t}_a = 1$. In shared-unary piecewise, rather than approximating the unary factors by $\tilde{t}_i = 1$, we incorporate them exactly, that is, we take $\tilde{t}_i(y_i) = t_i(y_i, \mathbf{x}, \theta)$ for the unary factors, and $\tilde{t}_a = 1$ otherwise. These messages are the result of one parallel BP iteration from uniform messages, so we call them *parallel BP(1) messages*. These messages yield the approximate partition function:

$$Z_{\text{PWU}} = \prod_{a \in NU} \left(\sum_{y_a} t_a(y_a, \mathbf{x}, \theta) \prod_{i \in a} t_i(y_i, \mathbf{x}, \theta) \right) \prod_i \left(\sum_{y_i} t_i(y_i, \mathbf{x}, \theta) \right)^{2-d_i}, \tag{5}$$

Iterations	
0	$\tilde{t}_a = 1$ for all a
1	$\tilde{t}_a = 1$ for nonunary a; $\tilde{t}_i = t_i$ for variables i
2	$\tilde{t}_a = \prod_{i \in a} m_{ai}^{(2)}(y_i)$, where $m_{ai}^{(2)}(y_i) = \sum_{y \setminus y_i} t_a(y_a, \mathbf{x}, \theta) \prod_{j \in N(a) \setminus i} t_j(y_j, \mathbf{x}, \theta)$

Table 1: Message settings after zero, one, and two parallel iterations of BP. Recall that the nonunary factors are assumed to be weakly canonical.

where NU is the set of nonunary factors in the model. We can distribute terms in Z_{PWU} to yield a form similar to standard piecewise. First, we define a normalized version of the unary factors as

$$p_i(y_i) = \frac{t_i(y_i, \mathbf{x}, \theta)}{\sum_{y_i'} t_i(y_i', \mathbf{x}, \theta)}. \tag{6}$$

Then we can distribute terms in (5) to obtain

$$Z_{\text{PWU}} = \prod_{a \in NU} \left(\sum_{y_a} t_a(y_a, \mathbf{x}, \theta) \prod_{i \in N(a)} p_i(y_i) \right) \prod_i \sum_{y_i} t_i(y_i, \mathbf{x}, \theta). \tag{7}$$

So shared-unary piecewise is the same as regular piecewise, except that we share normalized unary factors among all of the higher-way pieces that they neighbor. By normalizing the unary factors before spreading them across all the pieces, intuitively we avoid overcounting their sum.

5 Shared-Unary Pseudomarginals

In this section, we derive the pseudomarginals for shared-unary piecewise. Taking the derivative of $\log Z_{\text{PWU}}$ yields

$$\frac{\partial \log Z_{\text{PWU}}}{\partial \theta_i(y')} = \sum_{a \in NU(i)} \frac{\sum_{y_a} t_a(y_a, \mathbf{x}, \theta) \left(\prod_{j \in N(a) \setminus i} p_j(y_j) \right) \cdot \frac{\partial p_i(y_i)}{\partial \theta_i(y')}}{\sum_{y_a} t_a(y_a, \mathbf{x}, \theta) \left(\prod_{j \in N(a)} p_j(y_j) \right)} + p_i(y'), \tag{8}$$

where $NU(i)$ is set of all nonunary factors that neighbor variable i. Then substituting in

$$\frac{\partial p_i(y_i)}{\partial \theta_i(y')} = p(y_i)[1_{\{y' = y_i\}} - p_i(y')] \tag{9}$$

yields

$$\frac{\partial \log Z_{\text{PWU}}}{\partial \theta_i(y')} = p_i(y') + \left(\sum_{a \in NU(i)} \frac{\sum_{y_a \setminus y'} t_a(y_a, \mathbf{x}, \theta) \left(\prod_{j \in N(a)} p_j(y_j) \right)}{\sum_{y_a} t_a(y_a, \mathbf{x}, \theta) \left(\prod_{j \in N(a)} p_j(y_j) \right)} - p_i(y') \right) \tag{10}$$

$$= p_i(y') \left[1 + \left(\sum_{a \in NU(i)} C_a m_{ai}^{(2)}(y') - 1 \right) \right], \tag{11}$$

where $m_{ai}^{(2)}$ are the unnormalized BP messages after two parallel updates. We introduce the notation C_a to represent the denominator in (10), which is not the normalizing constant of $m_{ai}^{(2)}$.

Standard piecewise	
Neighborhood graph	node-split graph [5]
Pseudomarginal view	Messages at zero iterations
Belief view	$b_a(y_a) \propto t_a(y_a, \mathbf{x}, \theta)$
Shared-unary piecewise	
Neighborhood graph	node-split graph with pieces t_i for each variable i, and $\{t_a\} \cup \{p_i \mid i \in a\}$ for each nonunary a
Pseudomarginal view	Messages after one parallel iteration
Belief view	Summation-hack approximation to BP beliefs after two parallel iterations.
Cutout (one-step)	
Neighborhood graph	node-split graph with pieces G_a for each factor a, where G_a defined as in (14)
Pseudomarginal view	none
Belief view	BP beliefs after two parallel iterations

Table 2: Viewpoints on local training algorithms discussed in this note. For each method, "Neighborhood graph" means the graph on which the method can be viewed as performing exact maximum likelihood training. "Pseudomarginal view" lists the message setting with which the method approximates $\log Z$ by $\log Z_{\text{BP}}$. "Belief view" gives the beliefs with which the method can be viewed as approximating the gradient of the true likelihood. For reference, the BP messages after zero, one, and two parallel iterations are given in Table 1.

6 Cutout Method

The *cutout method* approximates the true gradient by performing exact inference on a subgraph. Each parameter is assigned a its own subgraph, but the subgraphs are allowed to overlap. Given a subgraph G_a for each factor a, we define the subgraph likelihood ℓ_a as the exact likelihood over the graph G_a. Let A be the set of factors in G_a and $t_A = \prod_{b \in A} t_b$. Then the subgraph likelihood can be written

$$\ell_a(\theta_a) = \frac{t_A(y_A, \mathbf{x}, \theta)}{\sum_{y'_A} t_A(y'_A, \mathbf{x}, \theta)} = \frac{t_A(y_A, \mathbf{x}, \theta)}{Z_A}. \tag{12}$$

Then the parameter vector θ is selected to solve the system of equations $\partial \ell_a / \partial \theta_a = 0$ for all a. In general, there does not exist a single objective function $\ell(\theta)$ whose partial derivatives match all of the $\partial \ell_a / \partial \theta_a$, because the vector field defined by $\partial \ell_a / \partial \theta_a$ has nonzero curl. In two dimensions, the curl of a vector field $F(x_1, x_2) = [f_1(x_1, x_2)\ f_2(x_1, x_2)]$ is given by

$$\operatorname{curl} F = \frac{\partial f_2}{\partial x_1} - \frac{\partial f_1}{\partial x_2}. \tag{13}$$

It is a standard theorem of vector calculus that a piecewise continuous vector field over $\mathbb{R}^n$ is the gradient of a function if and only if it has zero curl. To see this, observe that if F has nonzero curl, it is the gradient of a function f only if $\partial f / \partial x_1 x_2 \neq \partial f / \partial x_2 x_1$, which is impossible. The cutout vector field has nonzero curl essentialy because each θ_a is used in many ℓ_b, but only one ℓ_a is used to compute its approximate gradient.

In this note, we focus on a special case of the cutout method, the *one-step cutout method*. In one-step cutout, we choose G_a to be all of the neighboring factors of f_a, plus their unaries. That is, G_a is a factor graph with factors

$$A = \{t_b \,|\, \text{factor } t_b \text{ is distance 2 or less from factor } t_a\}. \tag{14}$$

(When counting distance between factors, we do not count variables, so that a path $a - i - b$ in the factor graph counts as one step.) In many situations, the cutout graph G_a is a tree, even when the original graph G is not, for example when G is a grid. If G_a is a tree, then we can compute $\partial \ell_a / \partial \theta_a$ exactly using two parallel iterations of BP on the original graph G. To see this, observe that because G_a is a tree of diameter 4, we can exactly compute Z_A by performing two parallel BP iterations on G_a. But the two-iteration messages on G_a are the same as the two-iteration messages on the original graph, which are given in Table 1.

The beliefs at the parallel BP(2) message setting are

$$b_a^{(2)}(y_a) \propto t_a(y_a, \mathbf{x}, \theta) \prod_{i \in N(i)} m_{ai}^{(2)}(y_i). \tag{15}$$

So from the belief viewpoint, one-step cutout approximates the ML gradient by substituting the beliefs (15) for the marginal probabilities.

7 Shared Unary as an Approximation to Cutout

Shared-unary piecewise can be viewed as an approximation to the cutout method. A general way of approximating a product of terms is the "summation hack":

$$\prod_i \epsilon_i = \prod_i 1 + (\epsilon_i - 1) \approx 1 + \left(\sum_i \epsilon_i - 1 \right), \tag{16}$$

where the approximation arises from a first-order Taylor expansion around $\vec{\epsilon} = 1$.

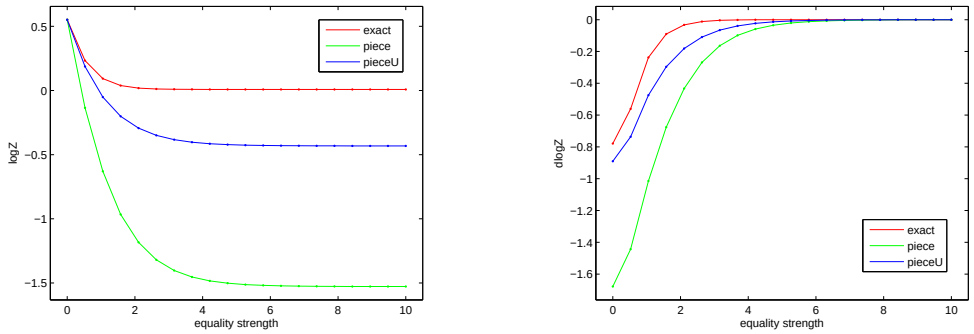

Figure 1: Comparison of piecewise and shared-unary piecewise approximations as a function of the equality strength s. Left, approximation to $\log Z$; right, approximation to its derivative.

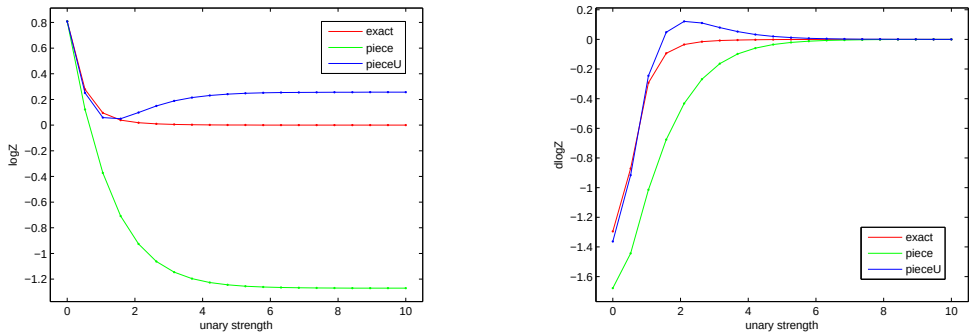

Figure 2: Approximation to $\log Z$ by piecewise and shared-unary piecewise as a function of the unary strength u.

Applying the summation hack to the one-step cutout beliefs (15), we obtain

$$b_i^{(2)}(y_i) \propto t_i(y_i, \mathbf{x}, \theta) \prod_{a \in NU(i)} C_a m_{ai}^{(2)}(y_i) \tag{17}$$

$$\approx t_i(y_i, \mathbf{x}, \theta) \left[1 + \left(\sum_{a \in NU(i)} C_a m_{ai}^{(2)}(y_i) - 1 \right) \right], \tag{18}$$

which are the same as the pseudomarginals (11) of Z_{PWU}, up to the proportionality constant of p_i. So we can view the shared-unary pseudomarginals either as the pseudomarginals after one BP iteration, or as an approximation to the beliefs after two iterations. This leads us to expect two sources of error in shared-unary piecewise: error may arise either from the summation hack, or because the model has long-distance interactions that cannot be propagated in two parallel BP iterations.

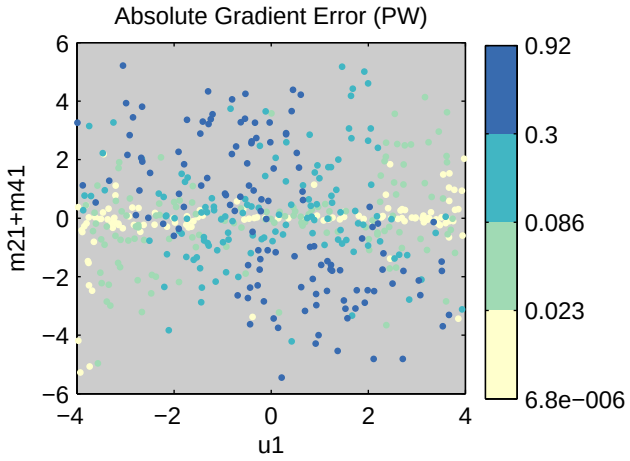

Figure 3: Absolute gradient error of standard piecewise as a function of the unary parameter and the sum of its incoming message strengths.

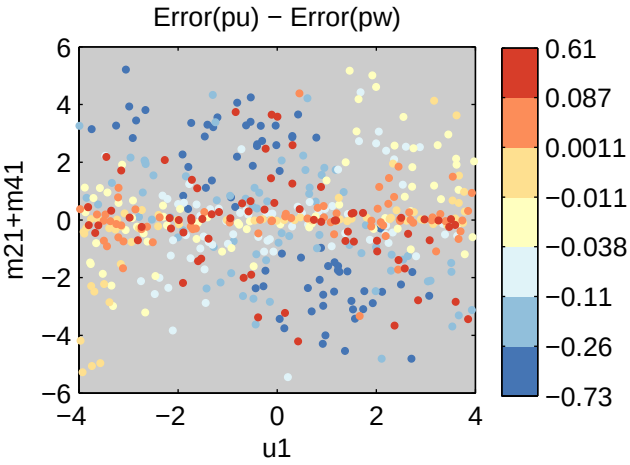

Figure 4: Difference in absolute gradient error between shared-unary piecewise and standard piecewise.

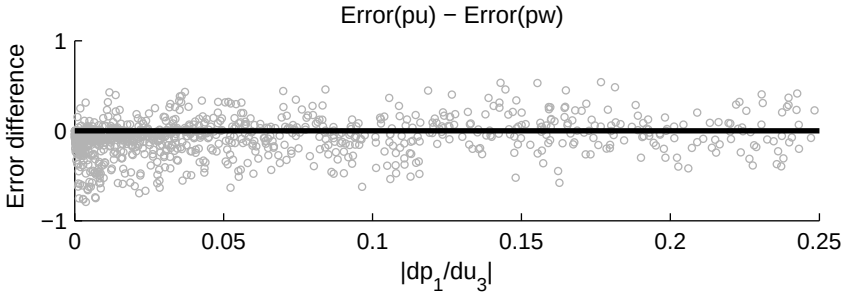

Figure 5: Difference in gradient error between shared-unary and standard piecewise. At left, when the magnitude of $\partial p(y_1)/\partial u_3$ is small, then shared-unary is superior. At right, where this derivative is large, shared-unary and standard piecewise are equivalent.

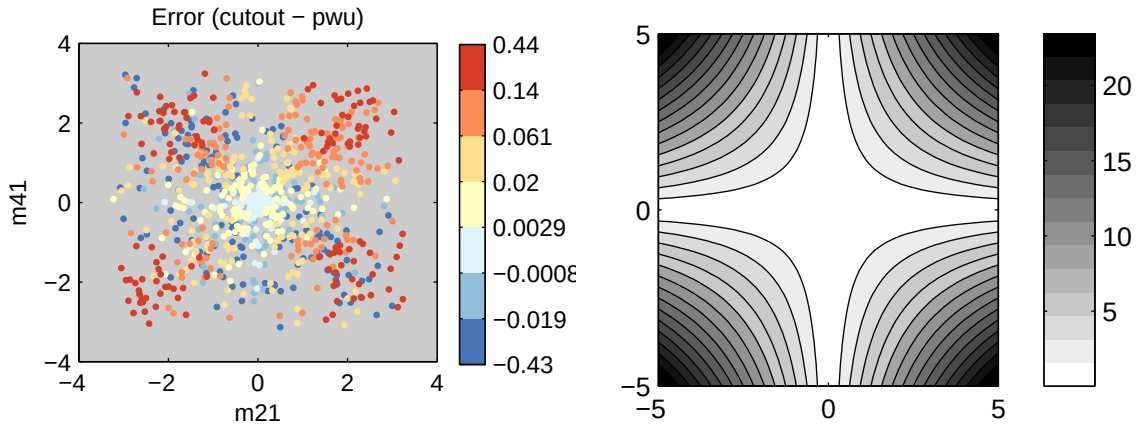

Figure 6: Difference in gradient error between cutout method and shared-unary piecewise as a function of the message strengths (left). At right, contours of the error of the "summation hack" Taylor expansion.

8 Simulations

These intuitions can be validated by simulation on a simple network. This data is generated from a three-node network of binary variables with pairwise factors

$$t_a(y_a) = \begin{pmatrix} 1 & e^{-s} \\ e^{-s} & 1 \end{pmatrix} \tag{19}$$

and unary factors $t_i(y_i) = [1 \;\; e^{-u}]$. We transform the pairwise factors into a three-variable exclusive-or factor times a unary factor, so that from the perspective of the learning algorithm, all the factors are unary. We focus on how the approximations to $\log Z$ and its derivatives change as a function of the model parameters. This is useful to study because the log likelihood equals $\log Z$ plus a linear function of the parameters, so examining $\log Z$ alone gives insight into how the approximation performs for any data set [8].

First, we look at single-dimensional plots of the approximate gradients, in which all of the unary parameters are tied, and we vary either the unary strength u or the equality strength s. As we vary the equality strength, for a fixed, strong unary strength of $e^{-u} = 0.2$, then shared-unary piecewise provides a much better approximation to $\log Z$ as a function of the equality strength s (Figure 1). As s approaches 0, the pairwise factors drop out, so that both the piecewise approximations are exact. In both Figures 1 and 2, we subtract $\log Z(0)$ from $\log Z_{\text{PW}}$. Without that correction, Z_{PW} is an upper bound but a strong overestimate. Also, in both figures, the plotted derivative is the negative of the pseudomarginal, because of the parameterization we use.

When we vary the unary strengths, on the other hand, shared unary has less desirable behavior (Figure 2). This figure shows the approximations to $\log Z$ and its derivative for a fixed equality strength $e^{-s} = 0.2$. Of course, as u approaches 0, the unary factors drop out, so that shared unary becomes equivalent to standard piecewise. Elsewhere, however, we see that shared-unary piecewise is no longer convex, because of the per-node normalization, and in fact we see a large regime where Z_{PWU} is increasing while the true Z is decreasing. Consequently, the derivative of Z_{PWU} crosses zero at several points when the exact objective does not, which is undesirable in an approximation. In other words, the piecewise pseudomarginal is sometimes negative.

We can get a more precise sense of when the piecewise approximations break down by examining their approximation error as a function of the incoming messages from the rest of the network. To do

this, we use a four-node Potts network of the form above, which is easier to interpret because there are no odd-length cycles. Also, it is helpful to leave the unary parameters untied, so that the model has five parameters $[s, u_1, u_2, u_3, u_4]$. We generate models by sampling uniformly over all parameter values in the range $[-4, 4]$. We measure the error in the pseudomarginal of y_1 for both standard and shared-unary piecewise. We plot the error in the pseudomarginal as a function of the message strengths m_{21} and m_{41} of the incoming messages to y_1 from its neighbors y_2 and y_4. By message strength, we mean the log ratio of the message value at 0 over the message value at 1.

For standard piecewise (Figure 3), we see as expected that the approximation error is greatest when the incoming messages are both strong and in disagreement with the local unary parameter. For shared unary, on the other hand, we report the difference in gradient error between shared-unary and standard piecewise (Figure 4). First, shared unary improves greatly over standard piecewise in the areas where piecewise performs worst, that is, when the incoming messages disagree strongly with the local unary. But shared unary is not always better than standard piecewise, especially when the messages are weak. This may be surprising because shared-unary performs an extra iteration of BP. However, the BP view of shared unary suggests two possible sources of error. First, one BP iteration can actually be worse than zero iterations, if nearby potentials contradict stronger factors elsewhere in the network. Second, shared unary may be a bad approximation to the two-iteration BP beliefs because of the summation hack.

We can isolate these two sources of error. First, to see where one iteration of BP might actually hurt, we look at the derivative of $p(y_1)$, the exact marginal probability of variable y_1, taken with respect to u_3, the unary parameter opposite y_1 in the graph. If this derivative has large magnitude, then we expect that the parameter u_3 has a large impact on the marginal $p(y_1)$, so that one iteration can make the beliefs worse if y_2 and y_4 have an effect in the opposite direction as y_3. In Figure 5, we show the error difference between shared-unary and standard piecewise as a function of $|\partial p(y_1)/\partial u_3|$. As this argument predicts, when this derivative has large magnitude, then the information from u_3, which neither method considers, is most important, so that neither method dominates. When this derivative has small magnitude, then the local information is more important, so shared-unary piecewise dominates.

Second, we can examine the effects of the summation hack by plotting the difference in gradient error between shared unary and cutout. The summation hack is accurate near the axes, so if both messages are strong, we expect shared unary to have high error. In Figure 6 it can be seen that shared unary performs worse than one-step cutout at exactly the places where the summation hack predicts.

An interesting observation here is that the cutout method is itself not always better than shared-unary piecewise, even though cutout performs an extra iteration of BP. This happens in cases when u_3 has large magnitude, so neither method can do well. In these cases, it can happen that the summation hack error pushes the shared-unary marginal in the direction of the correct marginal, so that shared unary performs better.

In summary, we have seen that in many situtions, shared-unary piecewise provides a better approximation to $\log Z$ and its derivatives than standard piecewise. The occasions when standard piecewise performs better than shared unary occur when there is strong influence from outside the pieces, in which case neither piecewise method is probably advisable. We have also demonstrated in simulation two potential sources of error in shared-unary piecewise: strong iteractions from outside the piece, and the summation hack. A potentially serious drawback to shared-unary piecewise, however, it is not convex in the parameters. Indeed, an important limitation of these simulations is that we have looked only at error in the gradient, not error in the optimal parameter settings, so we cannot assess to what extent the loss of convexity makes it harder to find good parameters.

Acknowledgements

We thank Martin Szummer for helpful conversations.

References

[1] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *J. Roy. Statist. Soc. Ser. B*, 36:192–236, 1974.

[2] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, (24):179–195, 1975.

[3] Tom Minka. Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research, 2005.

[4] Tom Minka. A survey of approximate CRF likelihoods. Microsoft Research Internal Memo, 2005.

[5] Charles Sutton. Node splitting view of piecewise training. Technical report, Microsoft Research, 2006.

[6] Charles Sutton and Andrew McCallum. Piecewise training of undirected models. In *21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[7] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006. To appear.

[8] Martin Szummer and Tom Minka. Analysis of extensions to piecewise training. Microsoft Research Internal Memo, in preparation, 2006.

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