Learning Graphical Model Parameters with Approximate Marginal Inference

Justin Domke

Abstract—Likelihood-based learning of graphical models faces challenges of computational complexity and robustness to model misspecification. This paper studies methods that fit parameters directly to maximize a measure of the accuracy of predicted marginals, taking into account both model and inference approximations at training time. Experiments on imaging problems suggest marginalization-based learning performs better than likelihood-based approximations on difficult problems where the model being fit is approximate in nature.

Index Terms—Graphical models, conditional random fields, machine learning, inference, segmentation

✦

1 INTRODUCTION

GRAPHICAL models are a standard tool in image processing, computer vision, and many other fields. In imaging, exact learning or inference is usually intractable due to the high treewidth of the graph.

Much previous work involves approximations of the likelihood (Section 4). In this paper, we suggest that parameter learning can instead be done using "marginalization-based" loss functions. These directly quantify the quality of the *predictions* of a given marginal inference algorithm. This has two major advantages. First, approximation errors in the inference algorithm are taken into account while learning. Second, this is robust to model misspecification.

The contributions of this paper are, first, the general framework of marginalization-based fitting as implicit differentiation. Second, we show that the parameter gradient can be computed by "perturbation"—that is, by rerunning the approximate algorithm twice, with the parameters perturbed slightly based on the current loss. Third, we introduce the strategy of "truncated fitting." Inference algorithms are based on optimization, where one iterates updates until some convergence threshold is reached. In truncated fitting, algorithms are derived to fit the marginals produced after a *fixed number* of updates, with no assumption of convergence. We show that this leads to significant speedups. We also derive a variant of this that can apply to likelihood-based learning. Finally, experimental results confirm that marginalization-based learning gives better results on difficult problems where inference approximations and model misspecification are most significant.

- *The author is with NICTA and Australia National University.*
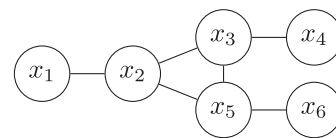 *E-mail: justin.domke@nicta.com.au.*

2 SETUP

2.1 Markov Random Fields (MRFs)

MRFs are probability distributions that may be written as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_c \psi(\mathbf{x}_c) \prod_i \psi(x_i). \qquad (1)$$

This is defined with reference to a graph, with one node for each random variable. The first product in (1) is over the set of *cliques* c in the graph, while the second is over all individual variables. For example, the graph

corresponds to the distribution

$$
\begin{aligned}
p(\mathbf{x}) = \frac{1}{Z} &\psi(x_1, x_2)\psi(x_2, x_3, x_5)\psi(x_3, x_4)\psi(x_5, x_6) \\
&\times \psi(x_1)\psi(x_2)\psi(x_3)\psi(x_4)\psi(x_5)\psi(x_6).
\end{aligned}
$$

Each function $\psi(\mathbf{x}_c)$ or $\psi(x_i)$ is positive, but otherwise arbitrary. The factor Z ensures normalization.

The motivation for these types of models is the Hammersley-Clifford theorem [1], which gives specific conditions under which a distribution can be written as in (1). Those conditions are that, first, each random variable is conditionally independent of all others, given its immediate neighbors, and, second, that each configuration $\mathbf{x}$ has nonzero probability. Often, domain knowledge about conditional independence can be used to build a reasonable graph, and the factorized representation in an MRF reduces the curse of dimensionality encountered in modeling a high-dimensional distribution.

2.2 Conditional Random Fields

One is often interested in modeling the conditional probability of $\mathbf{x}$, given observations $\mathbf{y}$. For such problems, it is natural to define a conditional random field [2]:

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{1}{Z(\mathbf{y})} \prod_c \psi(\mathbf{x}_c, \mathbf{y}) \prod_i \psi(x_i, \mathbf{y}).$$

Here, $\psi(\mathbf{x}_c, \mathbf{y})$ indicates that the value for a particular configuration $\mathbf{x}_c$ depends on the input $\mathbf{y}$. In practice, the form of this dependence is application dependent.

2.3 Inference Problems

Suppose we have some distribution $p(\mathbf{x} \mid \mathbf{y})$, we are given some input $\mathbf{y}$, and we need to guess a single output vector $\mathbf{x}^*$. What is the best guess?

The answer clearly depends on the meaning of "best." One framework for answering this question is the idea of a Bayes estimator [3]. One must specify some utility function $U(\mathbf{x}, \mathbf{x}')$, quantifying how "happy" one is to have guessed $\mathbf{x}$ if the true output is $\mathbf{x}'$. One then chooses $\mathbf{x}^*$ to maximize the expected utility:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_{\mathbf{x}'} p(\mathbf{x}' \mid \mathbf{y}) U(\mathbf{x}, \mathbf{x}').$$

One natural utility function is an indicator function, giving one for the exact value $\mathbf{x}'$, and zero otherwise. It is easy to show that for this utility, the optimal estimate is the popular maximum a posteriori (MAP) estimate.

Theorem. If $U(\mathbf{x}, \mathbf{x}') = I[\mathbf{x} = \mathbf{x}']$, then

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} p(\mathbf{x} \mid \mathbf{y}).$$

Little can be said, in general, about whether this utility function truly reflects user priorities. However, in high-dimensional applications, there are reasons for skepticism. First, the actual maximizing probability $p(\mathbf{x}^* \mid \mathbf{y})$ in an MAP estimate might be extremely small, so much so that astronomical numbers of examples might be necessary before one could expect to exactly predict the true output. Second, this utility does not distinguish between a prediction that contains only a single error at some component x_j, and one that is entirely wrong.

An alternative utility function, popular for imaging problems, quantifies the Hamming distance, or the *number of components* of the output vector that are correct. Maximizing this results in selecting the most likely value for each component independently.

Theorem. If $U(\mathbf{x}, \mathbf{x}') = \sum_i I[x_i = x_i']$, then

$$x_i^* = \arg\max_{x_i} p(x_i \mid \mathbf{y}). \tag{2}$$

This appears to have been originally called maximum posterior marginal (MPM) inference [4], though it has been reinvented under other names [5]. From a computational perspective, the main difficulty is not performing the trivial maximization in (2), but rather computing the marginals $p(x_i \mid \mathbf{y})$. The marginal-based loss functions introduced in Section 4.2 can be motivated by the idea that at test time, one will use an inference method similar to MPM where one is concerned only with the accuracy of the marginals.

The results of MAP and MPM inference will be similar if the distribution $p(\mathbf{x} \mid \mathbf{y})$ is heavily "peaked" at a single configuration $\mathbf{x}$. Roughly, the greater the entropy of $p(\mathbf{x} \mid \mathbf{y})$, the more there is to be gained in integrating over all possible configurations, as MPM does. A few papers have experimentally compared MAP and MPM inference [6], [7].

2.4 Exponential Family

The exponential family is defined by

$$p(\mathbf{x}; \boldsymbol{\theta}) = \exp(\boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{x}) - A(\boldsymbol{\theta})),$$

where $\boldsymbol{\theta}$ is a vector of parameters, $\mathbf{f}(\mathbf{x})$ is a vector of sufficient statistics, and the log-partition function

$$A(\boldsymbol{\theta}) = \log \sum_{\mathbf{x}} \exp \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{x}) \tag{3}$$

ensures normalization. Different sufficient statistics $\mathbf{f}(\mathbf{x})$ define different distributions. The exponential family is well understood in statistics. Accordingly, it is useful to note that an MRF (1) is a member of the exponential family, with sufficient statistics consisting of indicator functions for each possible configuration of each clique and each variable [8], namely,

$$\mathbf{f}(\mathbf{X}) = \{I[\mathbf{X}_c = \mathbf{x}_c] \mid \forall c, \mathbf{x}_c\} \cup \{I[X_i = x_i] \mid \forall i, x_i\}.$$

It is useful to introduce the notation $\theta(\mathbf{x}_c)$ to refer to the component of $\boldsymbol{\theta}$ corresponding to the indicator function $I[\mathbf{X}_c = \mathbf{x}_c]$, and similarly for $\theta(x_i)$. Then, the MRF in (1) would have $\psi(\mathbf{x}_c) = e^{\theta(\mathbf{x}_c)}$ and $\psi(x_i) = e^{\theta(x_i)}$. Many operations on graphical models can be more elegantly represented using this exponential family representation.

A standard problem in the exponential family is to compute the mean value of $\mathbf{f}$,

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \sum_{\mathbf{x}} p(\mathbf{x}; \boldsymbol{\theta}) \mathbf{f}(\mathbf{x}),$$

called the "mean parameters." It is easy to show that these are equal to the gradient of the log-partition function

$$\frac{dA}{d\boldsymbol{\theta}} = \boldsymbol{\mu}(\boldsymbol{\theta}). \tag{4}$$

For an exponential family corresponding to an MRF, computing $\boldsymbol{\mu}$ is equivalent to computing all the marginal probabilities. To see this, note that, using a similar notation for indexing $\boldsymbol{\mu}$ as for $\boldsymbol{\theta}$ above,

$$\boldsymbol{\mu}(\mathbf{x}_c; \boldsymbol{\theta}) = \sum_{\mathbf{X}} p(\mathbf{X}; \boldsymbol{\theta}) I[\mathbf{X}_c = \mathbf{x}_c] = p(\mathbf{x}_c; \boldsymbol{\theta}).$$

Conditional distributions can be represented by thinking of the parameter vector $\boldsymbol{\theta}(\mathbf{y}; \boldsymbol{\gamma})$ as being a function of the input $\mathbf{y}$, where $\boldsymbol{\gamma}$ are now the free parameters rather than $\boldsymbol{\theta}$. (Again, the nature of the dependence of $\boldsymbol{\theta}$ on $\mathbf{y}$ and $\boldsymbol{\gamma}$ will vary by application.) Then, we have that

$$p(\mathbf{x} \mid \mathbf{y}; \boldsymbol{\gamma}) = \exp(\boldsymbol{\theta}(\mathbf{y}; \boldsymbol{\gamma}) \cdot \mathbf{f}(\mathbf{x}) - A(\boldsymbol{\theta}(\mathbf{y}; \boldsymbol{\gamma}))), \tag{5}$$

sometimes called a curved conditional exponential family.

2.5 Learning

The focus of this paper is learning of model parameters from data. (Automatically determining graph *structure* remains an active research area, but is not considered here.) Specifically, we take the goal of learning to be to minimize the empirical risk

$$R(\boldsymbol{\theta}) = \sum_{\hat{\mathbf{x}}} L(\boldsymbol{\theta}, \hat{\mathbf{x}}), \qquad (6)$$

where the summation is over all examples $\hat{\mathbf{x}}$ in the dataset, and the loss function $L(\boldsymbol{\theta}, \hat{\mathbf{x}})$ quantifies how well the distribution defined by the parameter vector $\boldsymbol{\theta}$ matches the example $\hat{\mathbf{x}}$. Several loss functions are considered in Section 4.

We assume that the empirical risk will be fit by some gradient-based optimizations. Hence, the main technical issues in learning are which loss function to use and how to compute the gradient $\frac{dL}{d\boldsymbol{\theta}}$.

In practice, we will usually be interested in fitting conditional distributions. Using the notation from (5), we can write this as

$$R(\boldsymbol{\gamma}) = \sum_{(\hat{\mathbf{y}}, \hat{\mathbf{x}})} L(\boldsymbol{\theta}(\hat{\mathbf{y}}, \boldsymbol{\gamma}), \hat{\mathbf{x}}).$$

Note that if one has recovered $\frac{dL}{d\boldsymbol{\theta}}$, $\frac{dL}{d\boldsymbol{\gamma}}$ is immediate from the vector chain rule as

$$\frac{dL}{d\boldsymbol{\gamma}} = \frac{d\boldsymbol{\theta}^T}{d\boldsymbol{\gamma}} \frac{dL}{d\boldsymbol{\theta}}. \qquad (7)$$

Thus, the main technical problems involved in fitting a conditional distribution are similar to those for a generative distribution: One finds $\boldsymbol{\theta} = \boldsymbol{\theta}(\hat{\mathbf{y}}, \boldsymbol{\gamma})$, computes L and $\frac{dL}{d\boldsymbol{\theta}}$ on example $\hat{\mathbf{x}}$ exactly as in the generative case, and finally, recovers $\frac{dL}{d\boldsymbol{\gamma}}$ from (7). So, for simplicity, $\mathbf{y}$ and $\boldsymbol{\gamma}$ will largely be ignored in the theoretical developments below.

3 VARIATIONAL INFERENCE

This section reviews approximate methods for computing marginals, with notation based on Wainwright and Jordan [8]. For readability, all proofs in this section are presented in Appendix A, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.31.

The relationship between the marginals and the log-partition function in (4) is a key to defining approximate marginalization procedures. In Section 3.1, the exact variational principle shows that the (intractable) problem of computing the log-partition function can be converted to a (still intractable) optimization problem. To derive a tractable marginalization algorithm, one approximates this optimization, yielding some approximate log-partition function $\tilde{A}(\boldsymbol{\theta})$. The approximate marginals are then taken as the *exact* gradient of $\tilde{A}$.

We define the reverse mapping $\boldsymbol{\theta}(\boldsymbol{\mu})$ to return some parameter vectors that yield that marginals $\boldsymbol{\mu}$. While this will, in general, not be unique [8, Section 3.5.2], any two vectors that produce the same marginals $\boldsymbol{\mu}$ will also yield the same distribution, and so, $p(\mathbf{x}; \boldsymbol{\theta}(\boldsymbol{\mu}))$ is unambiguous.

3.1 Exact Variational Principle

Theorem (Exact variational principle). *The log-partition function can also be represented as*

$$A(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu}), \qquad (8)$$

where

$$\mathcal{M} = \{\boldsymbol{\mu}' : \exists \boldsymbol{\theta}, \boldsymbol{\mu}' = \boldsymbol{\mu}(\boldsymbol{\theta})\}$$

is the marginal polytope and

$$H(\boldsymbol{\mu}) = -\sum_{\mathbf{x}} p(\mathbf{x}; \boldsymbol{\theta}(\boldsymbol{\mu})) \log p(\mathbf{x}; \boldsymbol{\theta}(\boldsymbol{\mu}))$$

is the entropy.

In tree-like graphs, this optimization can be solved efficiently. In general graphs, however, it is intractable in two ways. First, the marginal polytope $\mathcal{M}$ becomes difficult to characterize. Second, the entropy is intractable to compute.

Applying Danskin's theorem to (8) yields that

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \frac{dA}{d\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu}). \qquad (9)$$

Thus, the partition function (8) and marginals (9) can both be obtained from solving the same optimization problem. This close relationship between the log-partition function and marginals is heavily used in the derivation of approximate marginalization algorithms. To compute approximate marginals, first, derive an approximate version of the optimization in (8). Next, take the exact gradient of this approximate partition function. This strategy is used in both of the approximate marginalization procedures considered here: mean field (MNF) and tree-reweighted (TRW) belief propagation.

3.2 Mean Field

The idea of MNF is to approximate the exact variational principle by replacing $\mathcal{M}$ with some tractable subset $\mathcal{F} \subset \mathcal{M}$ such that $\mathcal{F}$ is easy to characterize, and for any vector $\boldsymbol{\mu} \in \mathcal{F}$, we can exactly compute the entropy. To create such a set $\mathcal{F}$, instead of considering the set of mean vectors obtainable from *any* parameter vector (which characterizes $\mathcal{M}$), consider a subset of *tractable* parameter vectors. The simplest way to achieve this is to restrict consideration to parameter vectors $\boldsymbol{\theta}$ with $\theta(\mathbf{x}_c) = 0$ for all factors c:

$$\mathcal{F} = \{\boldsymbol{\mu}' : \exists \boldsymbol{\theta}, \boldsymbol{\mu}' = \boldsymbol{\mu}(\boldsymbol{\theta}), \ \forall c, \ \theta(\mathbf{x}_c) = 0\}.$$

It is not hard to see that this corresponds to the set of *fully factorized* distributions. Note also that this is (in non-tree-like graphs) a nonconvex set because it has the same convex hull as $\mathcal{M}$, but is a proper subset. So, the MNF partition function approximation is based on the optimization

$$\tilde{A}(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{F}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu}), \qquad (10)$$

with approximate marginals corresponding to the maximizing vector $\boldsymbol{\mu}$, i.e.,

$$\tilde{\boldsymbol{\mu}}(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\mu} \in \mathcal{F}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu}). \qquad (11)$$

Since this is maximizing the same objective as the exact variational principle, but under a more restricted constraint set, clearly $\tilde{A}(\boldsymbol{\theta}) \leq A(\boldsymbol{\theta})$.

Here, since the marginals are coming from a fully factorized distribution, the exact entropy is available as

$$H(\boldsymbol{\mu}) = -\sum_i \sum_{x_i} \mu(x_i) \log \mu(x_i). \qquad (12)$$

The strategy we use to perform the maximization in (10) is block-coordinate ascent. Namely, we pick a coordinate j, then set $\mu(x_j)$ to maximize the objective, leaving $\mu(x_i)$ fixed for all $i \neq j$. The next theorem formalizes this.

Theorem (MNF updates). *A local maximum of (10) can be reached by iterating the updates*

$$\mu(x_j) \leftarrow \frac{1}{Z} \exp\left(\theta(x_j) + \sum_{c:j\in c} \sum_{\mathbf{x}_{c\backslash j}} \theta(\mathbf{x}_c) \prod_{i\in c\backslash j} \mu(x_i) \right),$$

where Z is a normalizing factor ensuring that $\sum_{x_j} \mu(x_j) = 1$.

3.3 Tree-Reweighted Belief Propagation

Whereas MNF replaced the marginal polytope with a subset, TRW belief propagation replaces it with a superset, $\mathcal{L} \supset \mathcal{M}$. This clearly can only increase the value of the approximate log-partition function. However, a further approximation is needed as the entropy remains intractable to compute for an arbitrary mean vector $\boldsymbol{\mu}$. (It is not even defined for $\boldsymbol{\mu} \notin \mathcal{M}$.) Thus, TRW further approximates the entropy with a tractable upper bound. Taken together, these two approximations yield a tractable upper bound on the log-partition function.

Thus, TRW is based on the optimization problem

$$\tilde{A}(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu}\in\mathcal{L}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + \tilde{H}(\boldsymbol{\mu}). \qquad (13)$$

Again, the approximate marginals are simply the maximizing vector $\boldsymbol{\mu}$, i.e.,

$$\tilde{\boldsymbol{\mu}}(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\mu}\in\mathcal{L}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + \tilde{H}(\boldsymbol{\mu}). \qquad (14)$$

The relaxation of the local polytope used in TRW is the *local polytope*

$$\mathcal{L} = \left\{ \boldsymbol{\mu} : \sum_{\mathbf{x}_{c\backslash i}} \mu(\mathbf{x}_c) = \mu(x_i), \ \sum_{x_i} \mu(x_i) = 1 \right\}. \qquad (15)$$

Since any valid marginal vector must obey these constraints, clearly $\mathcal{M} \subset \mathcal{L}$. However, $\mathcal{L}$ in general also contains unrealizable vectors (though on trees $\mathcal{L} = \mathcal{M}$). Thus, the marginal vector returned by TRW may, in general, be inconsistent in the sense that no joint distribution yields those marginals.

The entropy approximation used by TRW is

$$\tilde{H}(\mu) = \sum_i H(\mu_i) - \sum_c \rho_c I(\mu_c), \qquad (16)$$

where $H(\mu_i) = -\sum_{x_i} \mu(x_i) \log \mu(x_i)$ is the univariate entropy corresponding to variable i, and

$$I(\mu_c) = \sum_{\mathbf{x}_c} \mu(\mathbf{x}_c) \log \frac{\mu(\mathbf{x}_c)}{\prod_{i\in c} \mu(x_i)} \qquad (17)$$

is the mutual information corresponding to the variables in the factor c. The motivation for this approximation is that if the constants ρ_c are selected appropriately, this gives an upper bound on the true entropy.

Theorem (TRW entropy bound). *Let $Pr(\mathcal{G})$ be a distribution over tree structured graphs and define $\rho_c = Pr(c \in \mathcal{G})$. Then, with $\tilde{H}$ as defined in (16),*

$$\tilde{H}(\boldsymbol{\mu}) \geq H(\boldsymbol{\mu}).$$

Thus, TRW is maximizing an upper bound on the exact variational principle, under an expanded constraint set. Since both of these changes can only increase the maximum value, we have that $\tilde{A}(\boldsymbol{\theta}) \geq A(\boldsymbol{\theta})$.

Now, we consider how to actually compute the approximate log-partition function and associated marginals. Consider the message-passing updates

$$m_c(x_i) \propto \sum_{\mathbf{x}_{c\backslash i}} e^{\frac{1}{\rho_c}\theta(\mathbf{x}_c)} \prod_{j\in c\backslash i} e^{\theta(x_j)} \frac{\prod_{d:j\in d} m_d(x_j)^{\rho_d}}{m_c(x_j)}, \qquad (18)$$

where "$\propto$" is used as an assignment operator to mean assigning after normalization.

Theorem (TRW updates). *Let ρ_c be as in the previous theorem. Then, if the updates in (18) reach a fixed point, the marginals defined by*

$$\mu(\mathbf{x}_c) \propto e^{\frac{1}{\rho_c}\theta(\mathbf{x}_c)} \prod_{i\in c} e^{\theta(x_i)} \frac{\prod_{d:i\in d} m_d(x_i)^{\rho_d}}{m_c(x_i)},$$

$$\mu(x_i) \propto e^{\theta(x_i)} \prod_{d:i\in d} m_d(x_i)^{\rho_d}$$

constitute the global optimum of (13).

So, if the updates happen to converge, we have the solution. Meltzer et al. [9] show that on certain graphs made up of *monotonic chains*, an appropriate ordering of messages does assure convergence. (The proof is essentially that under these circumstances, message passing is equivalent to coordinate ascent in the dual.)

TRW simplifies into loopy belief propagation by choosing $\rho_c = 1$ everywhere, though the bounding property is lost.

4 LOSS FUNCTIONS

For space, only a representative sample of prior work can be cited. A recent review [10] is more thorough.

Though, technically, a "loss" should be minimized, we continue to use this terminology for the likelihood and its approximations, where one wishes to maximize.

For simplicity, the discussion below is for the generative setting. Using the same loss functions for training a conditional model is simple (Section 2.5).

4.1 The Likelihood and Approximations

The classic loss function would be the likelihood, with

$$L(\boldsymbol{\theta}, \mathbf{x}) = \log p(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{x}) - A(\boldsymbol{\theta}). \qquad (19)$$

This has the gradient

$$\frac{dL}{d\boldsymbol{\theta}} = \mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}(\boldsymbol{\theta}). \qquad (20)$$

One argument for the likelihood is that it is efficient; given a correct model, as data increase, it converges to true parameters at an asymptotically optimal rate [11].

Some previous works use tree-structured graphs where marginals may be computed exactly [12]. Of course, in high-treewidth graphs, the likelihood and its gradient will be intractable to compute exactly due to the presence of the log-partition function $A(\boldsymbol{\theta})$ and marginals $\boldsymbol{\mu}(\boldsymbol{\theta})$. This has motivated a variety of approximations. The first is to approximate the marginals $\boldsymbol{\mu}$ using Markov chain Monte Carlo [13], [14]. This can lead to high computational expense (particularly in the conditional case, where different chains must be run for each input). Contrastive divergence [15] further approximates these samples by running the Markov chain for only a few steps, but starting at the data points [16]. If the Markov chain is run long enough, these approaches can give an arbitrarily good approximation. However, Markov chain parameters may need to be adjusted to the particular problem, and these approaches are generally slower than those discussed below.

4.1.1 Surrogate Likelihood

A seemingly heuristic approach would be to replace the marginals in (20) with those from an approximate inference method. This approximation can be quite principled if one thinks instead of approximating the log-partition function in the likelihood itself (19). Then, the corresponding approximate marginals will emerge as the *exact* gradient of this surrogate loss. This "surrogate likelihood" [17] approximation appears to be the most widely used loss in imaging problems, with marginals approximated by either MNF [18], [19], TRW [20], or LBP [21], [22], [23], [24], [25]. However, the terminology of "surrogate likelihood" is not widespread, and in most cases, only the gradient is computed, meaning the optimization cannot use line searches.

If one uses a log-partition approximation that provides a bound on the true log-partition function, the surrogate likelihood will then bound the true likelihood. Specifically, mean field-based surrogate likelihood is an upper bound on the true likelihood, while TRW-based surrogate likelihood is a lower bound.

4.1.2 Expectation Maximization

In many applications, only a subset of variables may be observed. Suppose that we want to model $\mathbf{x} = (\mathbf{z}, \mathbf{h})$ where $\mathbf{z}$ is observed, but $\mathbf{h}$ is hidden. A natural loss function here is the expected maximization (EM) loss

$$L(\boldsymbol{\theta}, \mathbf{z}) = \log p(\mathbf{z}; \boldsymbol{\theta}) = \log \sum_{\mathbf{h}} p(\mathbf{z}, \mathbf{h}; \boldsymbol{\theta}).$$

It is easy to show that this is equivalent to

$$L(\boldsymbol{\theta}, \mathbf{z}) = A(\boldsymbol{\theta}, \mathbf{z}) - A(\boldsymbol{\theta}), \tag{21}$$

where $A(\boldsymbol{\theta}, \mathbf{z}) = \log \sum_{\mathbf{h}} \exp \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{z}, \mathbf{h})$ is the log-partition function with $\mathbf{z}$ "clamped" to the observed values. If all variables are observed, $A(\boldsymbol{\theta}, \mathbf{z})$ reduces to $\boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{z})$.

On substituting a variational approximation for $A(\boldsymbol{\theta}, \mathbf{z})$, a "variational EM" algorithm [8, Section 6.2.2] can be recovered that alternates between computing approximate marginals and parameter updates. Here, because of the close relationship to the surrogate likelihood, we designate "surrogate EM" for the case where $A(\boldsymbol{\theta}, \mathbf{z})$ and

$A(\boldsymbol{\theta})$ may both be approximated, and the learning is done with a gradient-based method. To obtain a bound on the true EM loss, care is required. For example, lower bounding $A(\boldsymbol{\theta}, \mathbf{z})$ using MNF and upper bounding $A(\boldsymbol{\theta})$ using TRW means a lower bound on the true EM loss. However, using the same approximation for both $A(\boldsymbol{\theta})$ and $A(\boldsymbol{\theta}, \mathbf{z})$ appears to work well in practice [26].

4.1.3 Saddle-Point Approximation

A third approximation of the likelihood is to search for a "saddle point." Here, one approximates the gradient in (20) by running an (presumably approximate) MAP inference algorithm, and then imagining that the marginals put unit probability at the approximate MAP solution and zero elsewhere [27], [28], [21]. This is a heuristic method, but it can be expected to work well when the estimated MAP solution is close to the true MAP and the conditional distribution $p(\mathbf{x} \mid \mathbf{y})$ is strongly "peaked."

4.1.4 Pseudolikelihood

Finally, there are two classes of likelihood approximations that do not require inference. The first is the classic pseudolikelihood [29], where one uses

$$L(\boldsymbol{\theta}, \mathbf{x}) = \sum_i \log p(x_i \mid \mathbf{x}_{-i}; \boldsymbol{\theta}).$$

This can be computed efficiently, even in high treewidth graphs, since conditional probabilities are easy to compute. Besag [29] showed that, under certain conditions, this will converge to the true parameter vector as the amount of data becomes infinite. The pseudolikelihood has been used in many applications [30], [31]. Instead of the probability of individual variables given all others, one can take the probability of patches of variables given all others, sometimes called the "patch" pseudolikelihood [32]. This interpolates to the exact likelihood as the patches become larger, though some types of inferences are generally required.

4.1.5 Piecewise Likelihood

More recently, Sutton and McCallum [33] suggested the piecewise likelihood. The idea is to approximate the log-partition function as a sum of log-partition functions of the different "pieces" of the graph. There is flexibility in determining which pieces to use. In this paper, we will use pieces consisting of each clique and each variable, which worked better in practice than some alternatives. Then, one has the surrogate partition function

$$\tilde{A}(\boldsymbol{\theta}) = \sum_c A_c(\boldsymbol{\theta}) + \sum_i A_i(\boldsymbol{\theta}),$$
$$A_c(\boldsymbol{\theta}) = \log \sum_{\mathbf{x}_c} e^{\theta(\mathbf{x}_c)}, \quad A_i(\boldsymbol{\theta}) = \log \sum_{x_i} e^{\theta(x_i)}.$$

It is not too hard to show that $A(\boldsymbol{\theta}) \leq \tilde{A}(\boldsymbol{\theta})$. In practice, it is sometimes best to make some heuristic adjustments to the parameters after learning to improve test-time performance [34], [35].

4.2 Marginal-Based Loss Functions

Given the discussion in Section 4.1, one might conclude that the likelihood, while difficult to optimize, is an ideal loss function since, given a well-specified model, it will converge

to the true parameters at asymptotically efficient rates. However, this conclusion is complicated by two issues. First, of course, the maximum likelihood solution is computationally intractable, motivating the approximations above.

A second issue is that of *model misspecification*. For many types of complex phenomena, we will wish to fit a model that is approximate in nature. This could be true because the conditional independencies asserted by the graph do not exactly hold or because the parameterization of factors is too simplistic. These approximations might be made out of ignorance, due to a lack of knowledge about the domain being studied, or deliberately because the true model might have too many degrees of freedom to be fit with available data.

In the case of an approximate model, no "true" parameters exist. The idea of marginal-based loss functions is to instead consider how the model will be used. If one will compute marginals at test-time—perhaps for MPM inference (Section 2.3)—it makes sense to maximize the accuracy of these predictions. Further, if one will use an approximate inference algorithm, it makes sense to optimize the accuracy of the *approximate* marginals. This essentially fits into the paradigm of empirical risk minimization [36], [37]. The idea of training a probabilistic model using an alternative loss to the likelihood goes back at least to Bahl et al. [38] in the late 1980s.

There is reason to think the likelihood is somewhat robust to model misspecification. In the infinite data limit, it finds the "closest" solution in the sense of KL divergence since if q is the true distribution, then

$$KL(q\|p) = \text{const.} - \mathop{\mathbb{E}}_{q} \log p(\mathbf{x}; \boldsymbol{\theta}).$$

4.2.1 Univariate Logistic Loss

The univariate logistic loss [39] is defined by

$$L(\boldsymbol{\theta}, \mathbf{x}) = -\sum_i \log \mu(x_i; \boldsymbol{\theta}),$$

where we use the notation μ to indicate that the loss is implicitly defined with respect to the marginal predictions of some (possibly approximate) algorithms, rather than the true marginals. This measures the mean accuracy of all univariate marginals, rather than the joint distribution. This loss can be seen as empirical risk minimization of the KL divergence between the true marginals and the predicted ones since

$$\sum_i KL(q_i\|\mu_i) = \sum_i \sum_{x_i} q(x_i) \log \frac{q(x_i)}{\mu(x_i; \boldsymbol{\theta})}$$
$$= \text{const.} - \mathop{\mathbb{E}}_{q} \sum_i \log \mu(x_i; \boldsymbol{\theta}).$$

If defined on exact marginals, this is a type of composite likelihood [40].

4.2.2 Smoothed Univariate Classification Error

Perhaps the most natural loss in the conditional setting would be the univariate classification error:

$$L(\boldsymbol{\theta}, \mathbf{x}) = \sum_i S\left(\max_{x_i' \neq x_i} \mu(x_i'; \boldsymbol{\theta}) - \mu(x_i; \boldsymbol{\theta})\right),$$

where $S(\cdot)$ is the step function. This exactly measures the number of components of $\mathbf{x}$ that would be incorrectly predicted if using MPM inference. Of course, this loss is neither differentiable nor continuous, which makes it impractical to optimize using gradient-based methods. Instead Gross et al. [5] suggest approximating with a sigmoid function $S(t) = (1 + \exp(-\alpha t))^{-1}$, where α controls approximation quality.

There is evidence [36], [5] that the smoothed classification loss can yield parameters with lower univariate classification error under MPM inference. However, our experience is that it is also more prone to getting stuck in local minima, making experiments difficult to interpret. Thus, it is not included in the experiments below. Our experience with the univariate quadratic loss [41] is similar.

4.2.3 Clique Losses

Any of the above univariate losses can be instead taken based on cliques. For example, the clique logistic loss is

$$L(\boldsymbol{\theta}, \mathbf{x}) = -\sum_c \log \mu(\mathbf{x}_c; \boldsymbol{\theta}),$$

which may be seen as empirical risk minimization of the mean KL divergence of the true clique marginals to the predicted ones. An advantage of this with an exact model is consistency. Simple examples show cases where a model predicts perfect univariate marginals despite the joint distribution being very inaccurate. However, if all clique marginals are correct, the joint must be correct by the standard moment matching conditions for the exponential family [8].

4.2.4 Hidden Variables

Marginal-based loss functions can accommodate hidden variables by simply taking the sum in the loss over the *observed* variables only. A similar approach can be used with the pseudolikelihood or piecewise likelihood.

4.3 Comparison with Exact Inference

To compare the effects of different loss functions in the presence of model misspecification, this section contains a simple example where the graphical model takes the following "chain" structure:
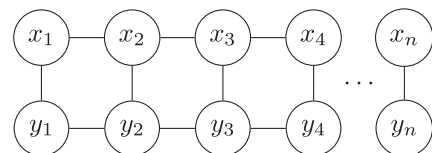
Here, exact inference is possible, so comparison is not complicated by approximate inference.

All variables are binary. Parameters are generated by taking $\theta(x_i)$ randomly from the interval $[-1, +1]$ for all i and x_i. Interaction parameters are taken as $\theta(x_i, x_j) = t$ when $x_i = x_j$, and $\theta(x_i, x_j) = -t$ when $x_i \neq x_j$, where t is randomly chosen from the interval $[-1, +1]$ for all (i, j). Interactions $\theta(y_i, y_j)$ and $\theta(x_i, y_i)$ are chosen in the same way.

To systematically study the effects of differing "amounts" of misspecification, after generating data we apply various circular shifts to $\mathbf{x}$. Thus, the data no longer correspond exactly to the structure of the graphical model being fit.
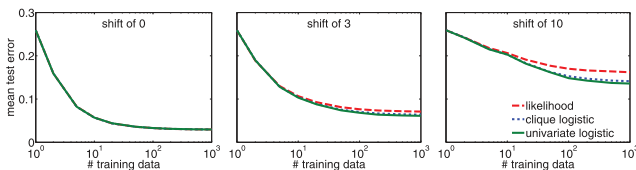
Fig. 1. Mean test error of various loss functions trained with exact inference. In the case of a well-specified model (shift of zero), the likelihood performs essentially identically to the marginal-based loss functions. However, when misspecification is introduced, quite different estimates result.

Thirty-two different random distributions were created. For each, various quantities of data were generated by Markov chain Monte Carlo, with shifts introduced after sampling. The likelihood was fit using the closed-form gradient (Section 4.1), while the logistic losses were trained using a gradient obtained via backpropagation (Section 7). Fig. 1 shows the mean test error (estimated on 1,000 examples), while Fig. 2 shows example marginals. We see that the performance of all methods deteriorates with misspecification, but the marginal-based loss functions are more resistant to these effects.

4.4 MAP-Based Training

Another class of methods explicitly optimizes the performance of MAP inference [42], [43], [44], [45], [25]. This paper focuses on applications that use marginal inference, and that may need to accommodate hidden variables, and so concentrates on likelihood and marginal-based losses.

5 IMPLICIT FITTING

We now turn to the issue of how to train high-treewidth graphical models to optimize the performance of a marginal-based loss function, based on some approximate inference algorithms. Now, computing the value of the loss for any of the marginal-based loss functions is not hard. One can simply run the inference algorithm and plug the resulting marginal into the loss. However, we also require the gradient $\frac{dL}{d\boldsymbol{\theta}}$.

Our first result is that the loss gradient can be obtained by solving a sparse linear system. Here, it is useful to introduce notation to distinguish the loss L, defined in terms of the parameters $\boldsymbol{\theta}$ from the loss Q, defined directly in terms of the marginals $\boldsymbol{\mu}$. (Note that though the notation suggests the application to marginal inference, this is a generic result.)

Theorem. *Suppose that*

$$\boldsymbol{\mu}(\boldsymbol{\theta}) := \underset{\boldsymbol{\mu}:B\boldsymbol{\mu}=\mathbf{d}}{\arg\max} \quad \boldsymbol{\theta} \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu}). \tag{22}$$

Define $L(\boldsymbol{\theta}, \mathbf{x}) = Q(\boldsymbol{\mu}(\boldsymbol{\theta}), \mathbf{x})$. *Then, letting* $D = \frac{d^2 H}{d\boldsymbol{\mu} d\boldsymbol{\mu}^T}$,

$$\frac{dL}{d\boldsymbol{\theta}} = \left(D^{-1}B^T (BD^{-1}B^T)^{-1} BD^{-1} - D^{-1} \right) \frac{dQ}{d\boldsymbol{\mu}}.$$

A proof may be found in Appendix B, available in the online supplemental material. This theorem states that, essentially, once one has computed the predicted marginals, the gradient of the loss with respect to marginals $\frac{dQ}{d\boldsymbol{\mu}}$ can be transformed into the gradient of the loss with respect to parameters $\frac{dL}{d\boldsymbol{\theta}}$ through the solution of a sparse linear system.

The optimization in (22) takes place under linear constraints, which encompasses the local polytope used in TRW message-passing (15). This theorem does not apply to MNF, as $\mathcal{F}$ is not a linear constraint set when viewed as a function of both clique and univariate marginals.

In any case, the methods developed below are simpler to use, as they do not require explicitly forming the constraint matrix B or solving the linear system.

6 PERTURBATION

This section observes that variational methods have a special structure that allows derivatives to be calculated
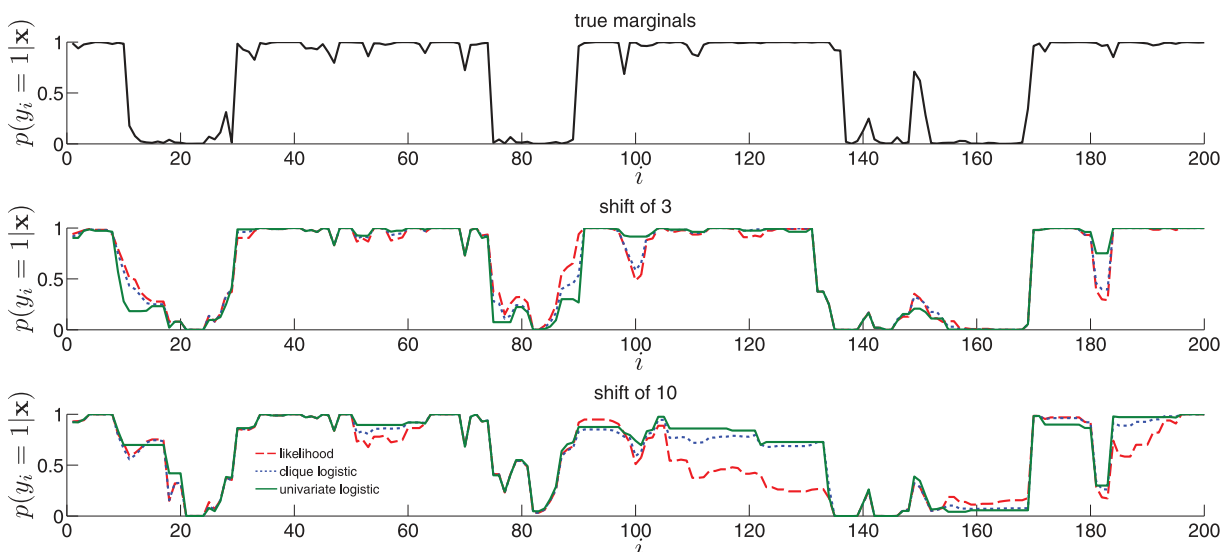

Fig. 2. Exact and predicted marginals for an example input. Predicted marginals are trained using 1,000 data. With low shifts, all loss functions lead to accurate predicted marginals. However, the univariate and clique logistic loss are more resistant to the effects of model misspecification. Legends as in Fig. 1.

without explicitly forming or inverting a linear system. We have, by the vector chain rule, that

$$\frac{dL}{d\boldsymbol{\theta}} = \frac{d\boldsymbol{\mu}^T}{d\boldsymbol{\theta}} \frac{dQ}{d\boldsymbol{\mu}}. \tag{23}$$

A classic trick in scientific computing is to efficiently compute Jacobian-vector products by finite differences. The basic result is that, for any vector $\mathbf{v}$,

$$\frac{d\boldsymbol{\mu}}{d\boldsymbol{\theta}^T} \mathbf{v} = \lim_{r \to 0} \frac{1}{r} (\boldsymbol{\mu}(\boldsymbol{\theta} + r\mathbf{v}) - \boldsymbol{\mu}(\boldsymbol{\theta})),$$

which is essentially just the definition of the derivative of $\boldsymbol{\mu}$ in the direction of $\mathbf{v}$. Now, this does not immediately seem helpful since (23) requires $\frac{d\boldsymbol{\mu}^T}{d\boldsymbol{\theta}}$, not $\frac{d\boldsymbol{\mu}}{d\boldsymbol{\theta}^T}$. However, with variational methods, these are symmetric. The simplest way to see this is to note that

$$\frac{d\boldsymbol{\mu}}{d\boldsymbol{\theta}^T} = \frac{d}{d\boldsymbol{\theta}^T} \left(\frac{dA}{d\boldsymbol{\theta}} \right) = \frac{dA}{d\boldsymbol{\theta} d\boldsymbol{\theta}^T}.$$

Domke [46] lists conditions for various classes of entropies that guarantee that A will be differentiable.

Combining the above three equations, the loss gradient is available as the limit

$$\frac{dL}{d\boldsymbol{\theta}} = \lim_{r \to 0} \frac{1}{r} \left(\boldsymbol{\mu} \left(\boldsymbol{\theta} + r \frac{dQ}{d\boldsymbol{\mu}} \right) - \boldsymbol{\mu}(\boldsymbol{\theta}) \right). \tag{24}$$

In practice, of course, the gradient is approximated using some finite r. The simplest approximation, one-sided differences, simply takes a single value of r in (24), rather than a limit. More accurate results at the cost of more calls to inference are given using two-sided differences, with

$$\frac{dL}{d\boldsymbol{\theta}} \approx \frac{1}{2r} \left(\boldsymbol{\mu} \left(\boldsymbol{\theta} + r \frac{dQ}{d\boldsymbol{\mu}} \right) - \boldsymbol{\mu} \left(\boldsymbol{\theta} - r \frac{dQ}{d\boldsymbol{\mu}} \right) \right),$$

which is accurate to order $o(r^2)$. Still more accurate results are obtained with "four-sided" differences, with

$$\frac{dL}{d\boldsymbol{\theta}} \approx \frac{1}{12r} \left(-\boldsymbol{\mu} \left(\boldsymbol{\theta} + 2r \frac{dQ}{d\boldsymbol{\mu}} \right) + 8\boldsymbol{\mu} \left(\boldsymbol{\theta} + r \frac{dQ}{d\boldsymbol{\mu}} \right) \right.$$
$$\left. - 8\boldsymbol{\mu} \left(\boldsymbol{\theta} - r \frac{dQ}{d\boldsymbol{\mu}} \right) + \boldsymbol{\mu} \left(\boldsymbol{\theta} - 2r \frac{dQ}{d\boldsymbol{\mu}} \right) \right),$$

which is accurate to order $o(r^4)$ [47].

Algorithm 1 shows more explicitly how the loss gradient could be calculated, using two-sided differences.

Algorithm 1. Calculating $\frac{dL}{d\boldsymbol{\theta}}$ by perturbation (two-sided).

1) Do inference. $\boldsymbol{\mu}^* \leftarrow \arg\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu})$

2) At $\boldsymbol{\mu}^*$, calculate the gradient $\frac{dQ}{d\boldsymbol{\mu}}$.

3) Calculate a perturbation size r.

4) Do inference on perturbed parameters.

   a) $\boldsymbol{\mu}^+ \leftarrow \arg\max_{\boldsymbol{\mu} \in \mathcal{M}} (\boldsymbol{\theta} + r\frac{dQ}{d\boldsymbol{\mu}}) \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu})$

   b) $\boldsymbol{\mu}^- \leftarrow \arg\max_{\boldsymbol{\mu} \in \mathcal{M}} (\boldsymbol{\theta} - r\frac{dQ}{d\boldsymbol{\mu}}) \cdot \boldsymbol{\mu} + H(\boldsymbol{\mu})$

5) Recover full derivative as $\frac{dL}{d\boldsymbol{\theta}} \leftarrow \frac{1}{2r}(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-)$.

The issue remains of how to calculate the step size r. Each of the approximations above becomes exact as $r \to 0$.

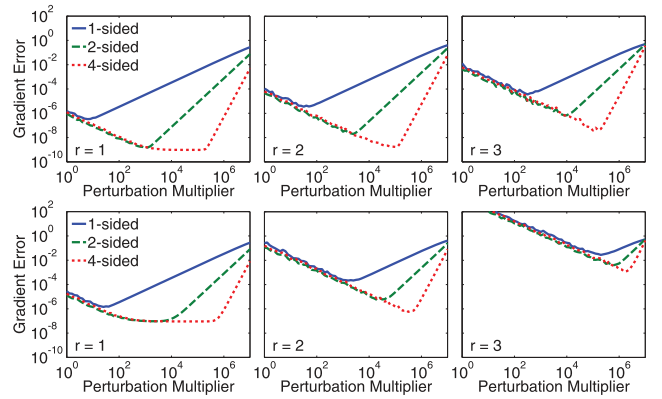
Fig. 3. An evaluation of perturbation multipliers m. Top: TRW. Bottom: MNF. Two effects are in play here: First, for too small a perturbation, numerical errors dominate. Meanwhile, for too large a perturbation, approximation errors dominate. We see that using two- or four-sided differences reduces approximation error, leading to better results with larger perturbations.

However, as r becomes very small, numerical error eventually dominates. To investigate this issue experimentally, we generated random models on a 10×10 binary grid, with each parameter $\theta(x_i)$ randomly chosen from a standard normal, while each interaction parameter $\theta(x_i, x_j)$ was chosen randomly from a normal with a standard deviation of s. In each case, a random value $\mathbf{x}$ was generated, and the "true" loss gradient was estimated by standard (inefficient) two-sided finite differences, with inference rerun after each component of $\boldsymbol{\theta}$ is perturbed independently. To this, we compare one-, two-, and four-sided perturbations. In all cases, the step size is, following Andrei [48], taken to be $r = m\epsilon^{\frac{1}{3}}(1 + \|\boldsymbol{\theta}\|_\infty)/\|\frac{dQ}{d\boldsymbol{\mu}}\|_\infty$, where ϵ is machine epsilon, and m is a multiplier that we will vary. Note that the optimal power of ϵ will depend on the finite difference scheme; $\frac{1}{3}$ is optimal for two-sided differences [49, Section 8.1]. All calculations take place in double precision with inference run until marginals changed by a threshold of less than 10^{-15}. Fig. 3 shows that using many-sided differences leads to more accuracy, at the cost of needing to run inference more times to estimate a single loss gradient. In the following experiments, we chose two-sided differences with a multiplier of 1 as a reasonable tradeoff between accuracy, simplicity, and computational expense.

Welling and Teh [50] used sensitivity of approximate beliefs to parameters to approximate joint probabilities of nonneighboring variables.

7 TRUNCATED FITTING

The previous methods for computing loss gradients are derived under the assumption that the inference optimization is solved exactly. In an implementation, of course, some convergence thresholds must be used.

Different convergence thresholds can be used in the learning stage and at test time. In practice, we have observed that too loose a threshold in the learning stage can lead to a bad estimated risk gradient, and learning terminating with a bad search direction. Meanwhile, a loose threshold can often be used at test time with few consequences. Usually, a difference of 10^{-3} in estimated

marginals has little practical impact, but this can still be enough to prevent learning from succeeding [51].

It seems odd that the learning algorithm would spend the majority of computational effort exploring tight convergence levels that are irrelevant to the practical performance of the model. Here, we *define* the learning objective in terms of the approximate marginals obtained after a fixed number of iterations. To understand this, one may think of the inference process not as an optimization, but rather as a large, nonlinear function. This clearly leads to a well-defined objective function. Inputting parameters, applying the iterations of either TRW or MNF, computing predicted marginals, and finally, a loss are all differentiable operations. Thus, the loss gradient is efficiently computable, at least in principle, by reverse-mode automatic differentiation (autodiff), an approach explored by Stovanov et al. [36], [52]. In preliminary work, we experimented with autodiff tools, but found these to be unsatisfactory for our applications for two reasons. First, these tools impose a computational penalty over manually derived gradients. Second, autodiff stores all intermediate calculations, leading to large memory requirements. The methods derived below use less memory, both in terms of constant factors and big-O complexity. Nevertheless, some of these problems are issues with current *implementations* of reverse-mode autodiff, avoidable in theory.

Both MNF and TRW involve steps where we first take a product of a set of terms, and then normalize. We define a "backnorm" operator, which is useful in taking derivatives over such operations, by

$$\text{backnorm}(\mathbf{g}, \mathbf{c}) = \mathbf{c} \odot (\mathbf{g} - \mathbf{g} \cdot \mathbf{c}).$$

This will be used in the algorithms here. More discussion on this point can be found in Appendix C, available in the online supplemental material.

7.1 Back Mean Field

The first backpropagating inference algorithm, back MNF, is shown as Algorithm 2. The idea is as follows: Suppose we start with uniform marginals, run N iterations of MNF, and then—regardless of if MNF has converged or not—take predicted marginals and plug them into one of the marginal-based loss functions. Since each step in this process is differentiable, this specifies the loss as a differentiable function of model parameters. We want the exact gradient of this function.

Algorithm 2. Back Mean Field
1) Initialize μ uniformly.
2) Repeat N times for all j:
    a) Push the marginals μ_j onto a stack.
    b) $\mu(x_j) \propto \exp\left(\theta(x_j) + \sum_{c:j \in c} \sum_{\mathbf{x}_{c\backslash j}} \theta(\mathbf{x}_c) \prod_{i \in c\backslash j} \mu(x_i) \right)$
3) Compute L, $\overleftarrow{\mu}(x_j) = \frac{dL}{d\mu(x_j)}$ and $\overleftarrow{\mu}(\mathbf{x}_c) = \frac{dL}{d\mu(\mathbf{x}_c)}$.
4) Initialize $\overleftarrow{\theta}(x_i) \leftarrow 0$, $\overleftarrow{\theta}(\mathbf{x}_c) \leftarrow 0$.
5) Repeat N times for all j (in reverse order):
    a) $\overleftarrow{\nu}_j \quad \leftarrow \text{backnorm}(\overleftarrow{\mu}_j, \mu_j)$
    b) $\overleftarrow{\theta}(x_j) \leftarrow \overleftarrow{\theta}(x_j) + \overleftarrow{\nu}(x_j)$
    c) $\overleftarrow{\theta}(\mathbf{x}_c) \leftarrow \overleftarrow{\theta}(\mathbf{x}_c) + \overleftarrow{\nu}(x_j) \prod_{i \in c\backslash j} \mu(x_i) \qquad \forall c : j \in c$

    d) $\overleftarrow{\mu}(x_i) \leftarrow \overleftarrow{\mu}(x_i) + \sum_{\mathbf{x}_{c\backslash i}} \overleftarrow{\nu}(x_j) \theta(\mathbf{x}_c) \prod_{k \in c\backslash\{i,j\}} \mu(x_k)$
                 $\forall c : j \in c, \forall i \in c\backslash j$
    e) Pull marginals μ_j from the stack.
    f) $\overleftarrow{\mu}_j(x_j) \leftarrow 0$

Theorem. *After execution of back MNF,*

$$\overleftarrow{\theta}(x_i) = \frac{dL}{d\theta(x_i)} \text{ and } \overleftarrow{\theta}(\mathbf{x}_c) = \frac{dL}{d\theta(\mathbf{x}_c)}.$$

A proof sketch is in Appendix C, available in the online supplemental material. Roughly speaking, the proof takes the form of a mechanical differentiation of each step of the inference process.

Note that, as written, back MNF only produces univariate marginals, and so cannot cope with loss functions making use of clique marginals. However, with MNF, the clique marginals are simply the product of univariate marginals: $\mu(\mathbf{x}_c) = \prod_{i \in c} \mu(x_i)$. Hence, any loss defined on clique marginals can equivalently be defined on univariate marginals.

7.2 Back TRW

Next, we consider truncated fitting with TRW inference. As above, we will assume that some fixed number N of inference iterations have been run, and we want to define and differentiate a loss defined on the current predicted marginals. Algorithm 3 shows the method.

Algorithm 3. Back TRW.
1) Initialize m uniformly.
2) Repeat N times for all pairs (c, i), with $i \in c$:
    a) Push the messages $m_c(x_i)$ onto a stack.

    b) $m_c(x_i) \propto \sum_{\mathbf{x}_{c\backslash i}} e^{\frac{1}{\rho_c}\theta(\mathbf{x}_c)} \prod_{j \in c\backslash i} e^{\theta(x_j)} \frac{\prod_{d:j \in d} m_d(x_j)^{\rho_d}}{m_c(x_j)}$

3) $\mu(\mathbf{x}_c) \propto e^{\frac{1}{\rho_c}\theta(\mathbf{x}_c)} \prod_{i \in c} e^{\theta(x_i)} \frac{\prod_{d:i \in d} m_d(x_i)^{\rho_d}}{m_c(x_i)} \qquad \forall c$

4) $\mu(x_i) \propto e^{\theta(x_i)} \prod_{d:i \in d} m_d(x_i)^{\rho_d} \qquad \forall i$

5) Compute L, $\overleftarrow{\mu}(x_i) = \frac{dL}{d\mu(x_i)}$ and $\overleftarrow{\mu}(\mathbf{x}_c) = \frac{dL}{d\mu(\mathbf{x}_c)}$.
6) For all c,
    a) $\overleftarrow{\nu}(\mathbf{x}_c) \leftarrow \text{backnorm}(\overleftarrow{\mu}_c, \mu_c)$
    b) $\overleftarrow{\theta}(\mathbf{x}_c) \overset{+}{\leftarrow} \frac{1}{\rho_c} \overleftarrow{\nu}(\mathbf{x}_c)$
    c) $\overleftarrow{\theta}(x_i) \overset{+}{\leftarrow} \sum_{\mathbf{x}_{c\backslash i}} \overleftarrow{\nu}(\mathbf{x}_c) \qquad\qquad \forall i \in c$
    d) $\overleftarrow{m_d}(x_i) \overset{+}{\leftarrow} \frac{\rho_d - I_{c=d}}{m_d(x_i)} \sum_{\mathbf{x}_{c\backslash i}} \overleftarrow{\nu} \qquad \forall i \in c, \forall d : i \in d$
7) For all i,
    a) $\overleftarrow{\nu}(x_i) \leftarrow \text{backnorm}(\overleftarrow{\mu}_i, \mu_i)$
    b) $\overleftarrow{\theta}(x_i) \overset{+}{\leftarrow} \overleftarrow{\nu}(x_i)$
    c) $\overleftarrow{m_d}(x_i) \overset{+}{\leftarrow} \rho_d \frac{\overleftarrow{\nu}(x_i)}{m_d(x_i)} \qquad\qquad \forall d : i \in d$
8) Repeat N times for all pairs (c, i) (in reverse order)

    a) $s(\mathbf{x}_c) \leftarrow e^{\frac{1}{\rho_c}\theta(\mathbf{x}_c)} \prod_{j \in c\backslash i} e^{\theta(x_j)} \frac{\prod_{d:j \in d} m_d(x_j)^{\rho_d}}{m_c(x_j)}$

    b) $\overleftarrow{\nu}(x_i) \leftarrow \text{backnorm}(\overleftarrow{m_{ci}}, m_{ci})$

    c) $\overleftarrow{\theta}(\mathbf{x}_c) \overset{+}{\leftarrow} \frac{1}{\rho_c} s(\mathbf{x}_c) \frac{\overleftarrow{\nu}(x_i)}{m_c(x_i)}$

    d) $\overleftarrow{\theta}(x_j) \overset{+}{\leftarrow} \sum_{\mathbf{x}_{c\backslash j}} s(\mathbf{x}_c) \frac{\overleftarrow{\nu}(x_i)}{m_c(x_i)} \qquad \forall j \in c\backslash i$

e) $\overleftarrow{m_d}(x_j) \overset{+}{\leftarrow} \frac{\rho_d - I_{c=d}}{m_d(x_j)} \sum_{\mathbf{x}_{c\backslash j}} s(\mathbf{x}_c) \frac{\overleftarrow{\nu}(x_i)}{m_c(x_i)}$

$$\forall j \in c\backslash i, \forall d : j \in d$$

f) Pull messages $m_c(x_i)$ from the stack.

g) $\overleftarrow{m_c}(x_i) \leftarrow 0$

Theorem. *After execution of back TRW,*

$$\overleftarrow{\theta}(x_i) = \frac{dL}{d\theta(x_i)} \ and \ \overleftarrow{\theta}(\mathbf{x}_c) = \frac{dL}{d\theta(\mathbf{x}_c)}.$$

Again, a proof sketch is in Appendix C, available in the online supplemental material.

If one uses pairwise factors only, uniform appearance probabilities of $\rho = 1$, removes all reference to the stack, and uses a convergence threshold in place of a fixed number of iterations, one obtains essentially Eaton and Ghahramani's [53, extended version, Fig. 5] back belief propagation. Here, we refer to the general strategy of using full (nontruncated) inference as "backpropagation," either with LBP, TRW, or MNF.

7.3 Truncated Likelihood and Truncated EM

Applying the truncated fitting strategies to any of the marginal-based loss functions is simple. Applying it to the likelihood or EM loss, however, is not so straightforward. The reason is that these losses ((19) and (21)) are defined, not in terms of predicted *marginals*, but in terms of *partition functions*. Nevertheless, we wish to compare to these losses in the experiments below. As we found truncation to be critical for speed, we instead derive a variant of truncated fitting.

The basic idea is to define a "truncated partition function." This is done by taking the predicted marginals, obtained after a fixed number of iterations, and plugging them into the entropy approximations used either for MNF (12) or TRW (16). The approximate entropy $\tilde{H}$ is then used in defining a truncated partition function as

$$\tilde{A}(\boldsymbol{\theta}) = \boldsymbol{\theta} \cdot \boldsymbol{\mu}(\boldsymbol{\theta}) - \tilde{H}(\boldsymbol{\mu}(\boldsymbol{\theta})).$$

As we will see below, with too few inference iterations, using this approximation can cause the surrogate likelihood to diverge. To see why, imagine an extreme case where *zero* inference iterations are used. This results in the loss $L(\boldsymbol{\theta}, \mathbf{x}) = \boldsymbol{\theta} \cdot (\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}^0) + \tilde{H}(\boldsymbol{\mu}^0)$, where $\boldsymbol{\mu}^0$ are the initial marginals. As long as the mean of $\mathbf{f}(\mathbf{x})$ over the dataset is not equal to $\boldsymbol{\mu}^0$, arbitrary loss can be achieved. With hidden variables, $A(\boldsymbol{\theta}, \mathbf{z})$ is defined similarly, but with the variables $\mathbf{z}$ "clamped" to the observed values. (Those variables will play no role in determining $\boldsymbol{\mu}(\boldsymbol{\theta})$).

8 EXPERIMENTS

These experiments consider three different datasets with varying complexity. In all cases, we try to keep the features used relatively simple. This means some sacrifice in performance, relative to using sophisticated features tuned more carefully to the different problem domains. However, given that our goal here is to gauge the relative performance of the different algorithms, we use simple features for the sake of experimental transparency.

We compare marginal-based learning methods to the surrogate likelihood/EM, the pseudolikelihood, and piecewise likelihood. These comparisons were chosen because, first, they are the most popular in the literature (Section 4). Second, the surrogate likelihood also requires marginal inference, meaning an "apples to apples" comparison using the same inference method. Third, these methods can all cope with hidden variables, which appear in our third dataset.

In each experiment, an "independent" model, trained using univariate features only with logistic loss was used to initialize others. The smoothed classification loss, because of more severe issues with local minima, was initialized using surrogate likelihood/EM.

8.1 Setup

All experiments here will be on vision problems, using a pairwise, four-connected grid. Learning uses the L-BFGS optimization algorithm. The values θ are linearly parameterized in terms of unary and edge features. Formally, we will fit two matrices, F and G, which determine all unary and edge features, respectively. These can be expressed most elegantly by introducing a bit more notation. Let $\boldsymbol{\theta}_i$ denote the set of parameter values $\theta(x_i)$ for all values x_i. If $\mathbf{u}(\mathbf{y}, i)$ denotes the vector of unary features for variable i given input image $\mathbf{y}$, then

$$\boldsymbol{\theta}_i = F\mathbf{u}(\mathbf{y}, i).$$

Similarly, let $\boldsymbol{\theta}_{ij}$ denote the set of parameter values $\theta(x_i, x_j)$ for all x_i, x_j. If $\mathbf{v}(\mathbf{y}, i, j)$ is the vector of edge features for pair (i, j), then

$$\boldsymbol{\theta}_{ij} = G\mathbf{v}(\mathbf{y}, i, j).$$

Once $\frac{dL}{d\boldsymbol{\theta}}$ has been calculated (for whichever loss and method), we can easily recover the gradients with respect to F and G by

$$\frac{dL}{dF} = \sum_i \frac{dL}{d\boldsymbol{\theta}_i} \mathbf{u}(\mathbf{y}, i)^T, \quad \frac{dL}{dG} = \sum_{ij} \frac{dL}{d\boldsymbol{\theta}_{ij}} \mathbf{v}(\mathbf{y}, i, j)^T.$$

8.2 Binary Denoising Data

In a first experiment, we create a binary denoising problem using the Berkeley segmentation dataset. Here, we took 132 200×300 images from the Berkeley segmentation dataset, binarized them according to if each pixel is above the image mean. The noisy input values are then generated as $y_i = x_i(1 - t_i^n) + (1 - x_i)t_i^n$, where x_i is the true binary label and $t_i \in [0, 1]$ is random. Here, $n \in (1, \infty)$ is the noise level, where lower values correspond to more noise. Thirty-two images were used for training and 100 for testing. This is something of a toy problem, but the ability to systematically vary the noise level is illustrative.

As unary features $\mathbf{u}(\mathbf{y}, i)$, we use only two features: a constant of 1 and the noisy input value at that pixel.

For edge features $\mathbf{v}(\mathbf{y}, i, j)$, we also use two features: one indicating that (i, j) is a horizontal edge and one indicating that (i, j) is a vertical edge. The effect is that vertical and horizontal edges have independent parameters.

For learning, we use full back TRW and back MNF (without message storing or truncation) for marginal-based loss functions, and the surrogate likelihood with the

TABLE 1
Binary Denoising Error Rates for Different Noise Levels n

Loss	$n = 1.25$ Train	$n = 1.25$ Test	$n = 1.5$ Train	$n = 1.5$ Test	$n = 5$ Train	$n = 5$ Test
surrogate likelihood	.149	.143	.103	.097	.031	.030
univariate logistic	.133	.126	.102	.096	.031	.030
clique logistic	.132	.126	.102	.096	.031	.030
univariate quad	.132	.126	.102	.096	.031	.030
smooth class. $\alpha = 5$	.136	.129	.105	.099	.032	.030
smooth class. $\alpha = 15$	.132	.126	.102	.096	.031	.030
smooth class. $\alpha = 50$	.132	.125	.102	.096	.030	.030
pseudo-likelihood	.207	.204	.117	.112	.032	.030
piecewise	.466	.481	.466	.481	.063	.058
independent	.421	.424	.367	.368	.129	.129

All methods use TRW inference with backpropagation-based learning with a threshold of 10^{-4}.

TABLE 2
Training and Test Errors on the Horses Dataset,
Using Either TRW on MNF Inference

Loss	λ	Mode	10 iters Train	10 iters Test	20 iters Train	20 iters Test	40 iters Train	40 iters Test
surrogate likelihood	10^{-3}	TRW	.421	.416	.088	.109	.088	.109
univariate logistic	10^{-3}	TRW	.065	.094	.064	.093	.064	.093
clique logistic	10^{-3}	TRW	.064	.094	.062	.093	.062	.092
sm. class. $\alpha = 5$	10^{-3}	TRW	.068	.097	.067	.097	.067	.096
sm. class. $\alpha = 15$	10^{-3}	TRW	.065	.097	.064	.096	.063	.096
sm. class. $\alpha = 50$	10^{-3}	TRW	.064	.096	.063	.095	.062	.095
surrogate likelihood	10^{-3}	MNF	.405	.383	.236	.226	.199	.200
univariate logistic	10^{-3}	MNF	.078	.108	.077	.106	.078	.106
clique logistic	10^{-3}	MNF	.073	.101	.075	.105	.079	.107
pseudolikelihood	10^{-4}	TRW					.222	.249
piecewise	10^{-4}	TRW					.202	.236
independent	10^{-4}						.095	.125

With too few iterations, the surrogate likelihood diverges.

gradient computed in the direct form (20). In all cases, a threshold on inference of 10^{-4} is used.

Error rates are shown in Table 1, while predicted marginals for an example test image are shown in Fig. 4. We compare against an independent model, which can be seen as truncated fitting with zero iterations or, equivalently, logistic regression at the pixel level. We see that for low noise levels, all methods perform well, while for high noise levels, the marginal-based losses outperform the surrogate likelihood and pseudolikelihood by a considerable margin. Our interpretation of this is that model misspecification is more pronounced with high noise, and other losses are less robust to this.
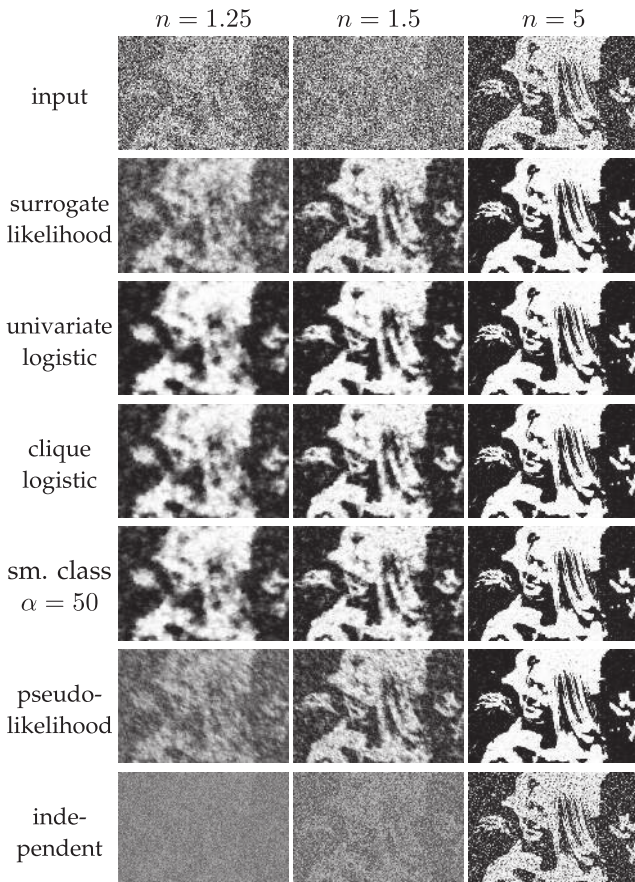

Fig. 4. Predicted marginals for an example binary denoising test image with different noise levels n.

8.3 Horses

Second, we use the Weizman horse dataset, consisting of 328 images of horses at various resolutions. We use 200 for training and 128 for testing. The set of possible labels x_i is again binary—either the pixel is part of a horse or not.

For unary features $\mathbf{u}(\mathbf{y}, i)$, we begin by computing the RGB intensities of each pixel, along with the normalized vertical and horizontal positions. We expand these five initial features into a larger set using sinusoidal expansion [54]. Specifically, denote the five original features by $\mathbf{s}$. Then, we include the features $\sin(\mathbf{c} \cdot \mathbf{s})$ and $\cos(\mathbf{c} \cdot \mathbf{s})$ for all binary vectors $\mathbf{c}$ of the appropriate length. This results in an expanded set of 64 features. To these we append a 36-component histogram of gradients [55], for a total of 100 features.

For edge features between i and j, we use a set of 21 "base" features: A constant of one, the l_2 norm of the difference of the RGB values at i and j, discretized as above 10 thresholds, and the maximum response of a Sobel edge filter at i or j, again discretized using 10 thresholds. To generate the final feature vector $\mathbf{v}(\mathbf{y}, i, j)$, this is increased into a set of 42 features. If (i, j) is a horizontal edge, the first half of these will contain the base feature, and the other half will be zero. If (i, j) is a vertical edge, the opposite situation occurs. This essentially allows for different parameterization of vertical and horizontal edges.

In a first experiment, we train models with truncated fitting with various numbers of iterations. The pseudolikelihood and piecewise likelihood use a convergence threshold of 10^{-5} for testing. Several trends are visible in Table 2. First, with less than 20 iterations, the truncated surrogate likelihood diverges and produces errors around 0.4. Second, TRW consistently outperforms MNF. Finally, marginal-based loss functions outperform the others, both in terms of training and test errors. Fig. 5 shows predicted marginals for an example test image. On this dataset, the pseudolikelihood, piecewise likelihood, and surrogate likelihood based on MNF are outperformed by an independent model where each label is predicted by input features independent of all others.
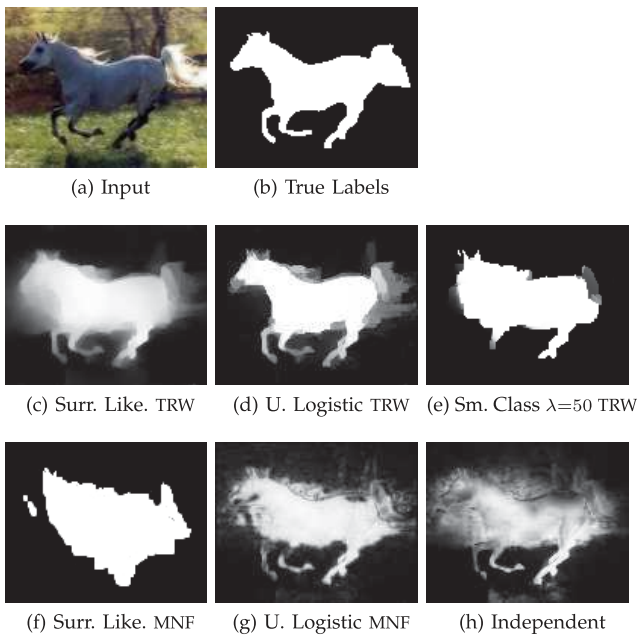
(a) Input   (b) True Labels

(c) Surr. Like. TRW   (d) U. Logistic TRW   (e) Sm. Class $\lambda{=}50$ TRW

(f) Surr. Like. MNF   (g) U. Logistic MNF   (h) Independent

Fig. 5. Predicted marginals for a test image from the horses dataset. Truncated learning uses 40 iterations.

8.4 Stanford Backgrounds Data

Our final experiments consider the Stanford backgrounds dataset. This consists of 715 images of resolution approximately 240×320. Most pixels are labeled as one of eight classes, with some unlabeled.

The unary features $\mathbf{u}(\mathbf{y}, i)$ we use here are identical to those for the horses dataset. In preliminary experiments, we tried training models with various resolutions. We found that reducing resolution to 20 percent of the original after computing features, then upsampling the predicted marginals yielded significantly better results than using the original resolution. Hence, this is done for all the following experiments. Edge features are identical to those for the horses dataset, except only based on the difference of RGB intensities, meaning 22 total edge features $\mathbf{v}(\mathbf{y}, i, j)$.

In a first experiment, we compare the performance of truncated fitting, perturbation, and backpropagation, using 100 images from this dataset for speed. We train with varying thresholds for perturbation and backpropagation, while for truncated learning we use various numbers of iterations. All models are trained with TRW to fit the univariate logistic loss. If a bad search direction is encountered, L-BFGS is reinitialized. Results are shown in

(a) Input Image   (b) True Labels

(c) Surrogate EM   (d) Univ. Logistic   (e) Clique Logistic

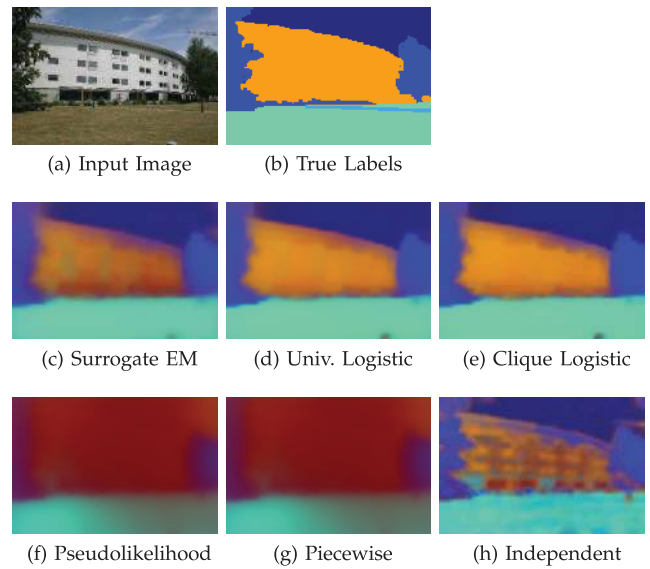(f) Pseudolikelihood   (g) Piecewise   (h) Independent

Fig. 6. Example marginals from the backgrounds dataset using 20 iterations for truncated fitting.

Fig. 7. We see that with loose thresholds, perturbation and backpropagation experience learning failure at suboptimal solutions. Truncated fitting is far more successful; using more iterations is slower to fit, but leads to better performance at convergence.

In a second experiment, we train on the entire dataset, with errors estimated using fivefold cross validation. Here, an incremental procedure was used, where first a subset of 32 images was trained on, with 1,000 learning iterations. The number of images was repeatedly doubled, with the number of learning iterations halved. In practice, this reduced training time substantially. Results are shown in Table 3 and Fig. 6. These results use a ridge regularization penalty of λ on all parameters. (This is relative to the empirical risk, as measured per pixel.) For EM and marginal-based loss functions, we set this as $\lambda = 10^{-3}$. We found in preliminary experiments that using a smaller regularization constant caused truncated EM to diverge even with 10 iterations. The pseudolikelihood and piecewise benefit from less regularization, and so, we use $\lambda = 10^{-4}$ there. Again, the marginal-based loss functions outperform others. In particular, they also perform quite well even with five iterations, where truncated EM diverges.

9 CONCLUSIONS

Training parameters of graphical models in a high treewidth setting involves several challenges. In this paper,

TABLE 3
Test Errors on the Backgrounds Dataset Using TRW Inference

		5 iters		10 iters		20 iters	
Loss	λ	Train	Test	Train	Test	Train	Test
surrogate EM	10^{-3}	.876	.877	.239	.249	.238	.248
univariate logistic	10^{-3}	.210	.228	.202	.224	.201	.223
clique logistic	10^{-3}	.206	.226	.198	.223	.195	.221
pseudolikelihood	10^{-4}					.516	.519
piecewise	10^{-4}					.335	.341
independent	10^{-4}					.293	.299

With too few iterations, surrogate EM diverges, leading to very high error rates.
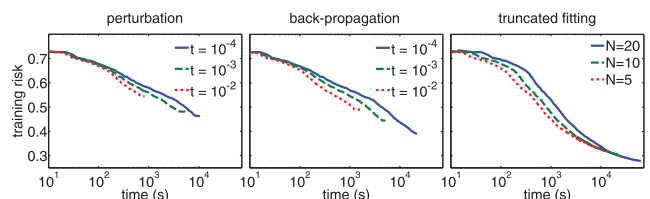

Fig. 7. Comparison of different learning methods on the backgrounds dataset with 100 images. All use an eight-core 2.26 GHz PC.

we focus on three: model misspecification, the necessity of approximate inference, and computational complexity.

The main technical contribution of this paper is several methods for training based on the marginals predicted by a given approximate inference algorithm. These methods take into account model misspecification and inference approximations. To combat computational complexity, we introduce "truncated" learning, where the inference algorithm only needs to be run for a fixed number of iterations. Truncation can also be applied, somewhat heuristically, to the surrogate likelihood.

Among previous methods, we experimentally find the surrogate likelihood to outperform the pseudolikelihood or piecewise learning. By more closely reflecting the test criterion of Hamming loss, marginal-based loss functions perform still better, particularly on harder problems (though the surrogate likelihood generally displays smaller train/test gaps). Additionally, marginal-based loss functions are more amenable to truncation, as the surrogate likelihood diverges with too few iterations.

REFERENCES

[1] J. Besag, "Spatial Interaction and the Statistical Analysis of Lattice Systems," *J. Royal Statistical Soc., Series B (Methodological)*, vol. 36, no. 2, pp. 192-236, 1974.

[2] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," *Proc. Int'l Conf. Machine Learning*, 2001.

[3] M. Nikolova, "Model Distortions in Bayesian MAP Reconstruction," *Inverse Problems and Imaging*, vol. 1, no. 2, pp. 399-422, 2007.

[4] J. Marroquin, S. Mitter, and T. Poggio, "Probabilistic Solution of Ill-Posed Problems in Computational Vision," *J. Am. Statistical Assoc.*, vol. 82, no. 397, pp. 76-89, 1987.

[5] S.S. Gross, O. Russakovsky, C.B. Do, and S. Batzoglou, "Training Conditional Random Fields for Maximum Labelwise Accuracy," *Proc. Advances in Neural Information Processing Systems*, 2007.

[6] S. Kumar, J. August, and M. Hebert, "Exploiting Inference for Approximate Parameter Learning in Discriminative Fields: An Empirical Study," *Proc. Int'l Conf. Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2005.

[7] P. Kohli and P. Torr, "Measuring Uncertainty in Graph Cut Solutions," *Computer Vision and Image Understanding*, vol. 112, no. 1, pp. 30-38, 2008.

[8] M. Wainwright and M. Jordan, "Graphical Models, Exponential Families, and Variational Inference," *Foundations and Trends in Machine Learning*, vol. 1, nos. 1/2, pp. 1-305, 2008.

[9] T. Meltzer, A. Globerson, and Y. Weiss, "Convergent Message Passing Algorithms—A Unifying View," *Proc. Conf. Uncertainty in Artificial Intelligence*, 2009.

[10] S. Nowozin and C.H. Lampert, "Structured Learning and Prediction in Computer Vision," *Foundations and Trends in Computer Graphics and Vision*, vol. 6, pp. 185-365, 2011.

[11] H. Cramér, *Mathematical Methods of Statistics*. Princeton Univ. Press, 1999.

[12] S. Nowozin, P.V. Gehler, and C.H. Lampert, "On Parameter Learning in CRF-Based Approaches to Object Class Image Segmentation," *Proc. European Conf. Computer Vision*, 2010.

[13] L. Stewart, X. He, and R.S. Zemel, "Learning Flexible Features for Conditional Random Fields," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1415-1426, Aug. 2008.

[14] C. Geyer, "Markov Chain Monte Carlo Maximum Likelihood," *Proc. Symp. Interface*, 1991.

[15] M. Carreira-Perpinan and G. Hinton, "On Contrastive Divergence Learning," *Proc. Int'l Workshop Artificial Intelligence and Statistics*, 2005.

[16] S. Roth and M.J. Black, "Fields of Experts," *Int'l J. Computer Vision*, vol. 82, no. 2, pp. 205-229, 2009.

[17] M.J. Wainwright, "Estimating the 'Wrong' Graphical Model: Benefits in the Computation-Limited Setting," *J. Machine Learning Research*, vol. 7, pp. 1829-1859, 2006.

[18] J.J. Weinman, L.C. Tran, and C.J. Pal, "Efficiently Learning Random Fields for Stereo Vision with Sparse Message Passing," *Proc. European Conf. Computer Vision*, pp. 617-630, 2008.

[19] T. Toyoda and O. Hasegawa, "Random Field Model for Integration of Local Information and Global Information," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1483-1489, Aug. 2008.

[20] A. Levin and Y. Weiss, "Learning to Combine Bottom-Up and Top-Down Segmentation," *Int'l J. Computer Vision*, vol. 81, no. 1, pp. 105-118, 2009.

[21] S. Kumar, J. August, and M. Hebert, "Exploiting Inference for Approximate Parameter Learning in Discriminative Fields: An Empirical Study," *Proc. Int'l Conf. Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2005.

[22] X. Ren, C. Fowlkes, and J. Malik, "Figure/Ground Assignment in Natural Images," *Proc. European Conf. Computer Vision*, 2006.

[23] S.V.N. Vishwanathan, N.N. Schraudolph, M.W. Schmidt, and K.P. Murphy, "Accelerated Training of Conditional Random Fields with Stochastic Gradient Methods," *Proc. Int'l Conf. Machine Learning*, 2006.

[24] X. Ren, C. Fowlkes, and J. Malik, "Learning Probabilistic Models for Contour Completion in Natural Images," *Int'l J. Computer Vision*, vol. 77, nos. 1-3, pp. 47-63, 2008.

[25] J. Yuan, J. Li, and B. Zhang, "Scene Understanding with Discriminative Structured Prediction," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[26] J.J. Verbeek and B. Triggs, "Scene Segmentation with CRFs Learned from Partially Labeled Images," *Proc. Advances in Neural Information Processing Systems*, 2007.

[27] D. Scharstein and C. Pal, "Learning Conditional Random Fields for Stereo," *Proc. IEEE Conf. Computer Vision and Pattern Recognition* , 2007.

[28] P. Zhong and R. Wang, "Using Combination of Statistical Models and Multilevel Structural Information for Detecting Urban Areas from a Single Gray-Level Image," *IEEE Trans. Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1469-1482, May 2007.

[29] J. Besag, "Statistical Analysis of Non-Lattice Data," *J. Royal Statistical Soc. Series D (The Statistician)*, vol. 24, no. 3, pp. 179-195, 1975.

[30] X. He, R.S. Zemel, and M.Á. Carreira-Perpiñán, "Multiscale Conditional Random Fields for Image Labeling," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004.

[31] S. Kumar and M. Hebert, "Discriminative Random Fields," *Int'l J. Computer Vision*, vol. 68, no. 2, pp. 179-201, 2006.

[32] S.C. Zhu and X. Liu, "Learning in Gibbsian Fields: How Accurate and How Fast Can It Be?" *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 1001-1006, June 2002.

[33] C. Sutton and A. McCallum, "Piecewise Training for Undirected Models," *Proc. Conf. Uncertainty Artificial Intelligence*, 2005.

[34] S. Kim and I.-S. Kweon, "Robust Model-Based Scene Interpretation by Multilayered Context Information," *Computer Vision and Image Understanding*, vol. 105, no. 3, pp. 167-187, 2007.

[35] J. Shotton, J.M. Winn, C. Rother, and A. Criminisi, "Textonboost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context," *Int'l J. Computer Vision*, vol. 81, no. 1, pp. 2-23, 2009.

[36] V. Stoyanov, A. Ropson, and J. Eisner, "Empirical Risk Minimization of Graphical Model Parameters Given Approximate Inference, Decoding, and Model Structure," *Proc. Int'l Workshop Artificial Intelligence and Statistics*, 2011.

[37] J. Domke, "Learning Convex Inference of Marginals," *Proc. Conf. Uncertainty Artificial Intelligence*, 2008.

[38] L.R. Bahl, P.F. Bron, P.V. de Souza, and R.L. Mercer, "A New Algorithm for the Estimation of Hidden Markov Model Parameters," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, 1988.

[39] S. Kakade, Y.W. Teh, and S. Roweis, "An Alternate Objective Function for Markovian Fields," *Proc. Int'l Conf. Machine Learning*, 2002.

[40] B.G. Lindsay, "Composite Likelihood Methods," *Contemporary Math.*, vol. 80, pp. 221-239, 1988.

[41] J. Domke, "Learning Convex Inference of Marginals," *Proc. Conf. Uncertainty Artificial Intelligence*, 2008.

[42] C. Desai, D. Ramanan, and C.C. Fowlkes, "Discriminative Models for Multi-Class Object Layout," *Int'l J. Computer Vision*, vol. 95, no. 1, pp. 1-12, 2011.

[43] M. Szummer, P. Kohli, and D. Hoiem, "Learning CRFs Using Graph Cuts," *Proc. European Conf. Computer Vision*, 2008.

[44] J.J. McAuley, T.E. de Campos, G. Csurka, and F. Perronnin, "Hierarchical Image-Region Labeling via Structured Learning," *Proc. British Machine Vision Conf.*, 2009.

[45] W. Yang, B. Triggs, D. Dai, and G.-S. Xia, "Scene Segmentation via Low-Dimensional Semantic Representation and Conditional Random Field," technical report, HAL, 2009.

[46] J. Domke, "Implicit Differentiation by Perturbation," *Proc. Advances in Neural Information Processing Systems*, 2010.

[47] A. Boresi and K. Chong, *Approximate Solution Methods in Eng. Mechanics.* Elsevier Science Inc., 1991.

[48] N. Andrei, "Accelerated Conjugate Gradient Algorithm with Finite Difference Hessian/Vector Product Approximation for Unconstrained Optimization," *J. Computational Applied Math.*, vol. 230, no. 2, pp. 570-582, 2009.

[49] J. Nocedal and S.J. Wright, *Numerical Optimization,* second ed. Springer, 2006.

[50] M. Welling and Y.W. Teh, "Linear Response Algorithms for Approximate Inference in Graphical Models," *Neural Computation,* vol. 16, pp. 197-221, 2004.

[51] J. Domke, "Parameter Learning with Truncated Message-Passing," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2011.

[52] V. Stoyanov and J. Eisner, "Minimum-Risk Training of Approximate CRF-Based NLP Systems," *Proc. Conf. North Am. Chapter of the Assoc. for Computational Linguistics: Human Language Technologies,* 2012.

[53] F. Eaton and Z. Ghahramani, "Choosing a Variable to Clamp," *Proc. Int'l Conf. Artificial Intelligence and Statistics,* 2009.

[54] G. Konidaris, S. Osentoski, and P. Thomas, "Value Function Approximation in Reinforcement Learning Using the Fourier Basis," *Proc. Conf. Artificial Intelligence,* 2011.

[55] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

Justin Domke received the PhD degree in computer science from the University of Maryland, College Park, in 2009. From 2009 to 2012, he was an assistant professor at the Rochester Institute of Technology. Since 2012, he has been a member of the Machine Learning group at NICTA.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.