Revisiting Uncertainty in Graph Cut Solutions

Daniel Tarlow
Dept. of Computer Science
University of Toronto
dtarlow@cs.toronto.edu

Ryan P. Adams
School of Engineering and Applied Sciences
Harvard University
rpa@seas.harvard.edu

Abstract

Graph cuts is a popular algorithm for finding the MAP assignment of many large-scale graphical models that are common in computer vision. While graph cuts is powerful, it does not provide information about the marginal probabilities associated with the solution it finds. To assess uncertainty, we are forced to fall back on less efficient and inexact inference algorithms such as loopy belief propagation, or use less principled surrogate representations of uncertainty such as the min-marginal approach of Kohli & Torr [8].

In this work, we give new justification for using min-marginals to compute the uncertainty in conditional random fields, framing the min-marginal outputs as exact marginals under a specially-chosen generative probabilistic model. We leverage this view to learn properly calibrated marginal probabilities as the result of straightforward maximization of the training likelihood, showing that the necessary subgradients can be computed efficiently using dynamic graph cut operations. We also show how this approach can be extended to compute multi-label marginal distributions, where again dynamic graph cuts enable efficient marginal inference and maximum likelihood learning. We demonstrate empirically that — after proper training — uncertainties based on min-marginals provide better-calibrated probabilities than baselines and that these distributions can be exploited in a decision-theoretic way for improved segmentation in low-level vision.

1. Introduction

Queries on random fields can be broadly classified into two types: queries for an optimum (finding a mode), and queries for a sum or integral (marginalization). In the first case, one might ask for the most likely joint configuration of the entire field. In the second class, one might ask for the marginal probability of a single variable taking some assignment. At first glance, these two types of queries may appear computationally similar; indeed, on a tree-structured graphical model they take the same amount of time. However, for some model classes there is a large discrepancy between the computational complexities of these queries. For example, when a graphical model is constrained to have binary variables and submodular interactions, the mode can be found in polynomial time using the graph cuts algorithm, while marginalization is #P-complete [7].

In computer vision, this discrepancy has contributed to a proliferation of optimization procedures centered around the graph cuts algorithm. Graph cuts are used both as a stand-alone procedure and as subroutine for algorithms such as alpha expansion [2], the min-marginal uncertainty of [8], the message passing of [5], and the Quadratic Pseudo Boolean Optimization algorithm [9]. Particularly given the efficient, freely available implementation of [1], graph cuts could be considered one of the most practical and powerful algorithms for inference in graphical models that is available to the computer vision practitioner.

Despite the successes of graph cuts, the algorithm is of limited applicability to queries of the second broad type. When viewed as a method for approximating the marginal probability of a variable in a graphical model, we show in the supplementary material that the min-marginal uncertainty of [8] can be off by a factor that is exponentially large in the number of variables in the model, and we show empirically that learning using this method as approximate inference can lead to poorly calibrated estimates of marginal probabilities. Marginal probabilities are important in many applications, including interactive segmentation, active learning, and multilabel image segmentation. They are perhaps even more important in low-level vision tasks, as random field models are often only the first component of a larger computer vision system. In this respect, it is desirable to be able to provide higher-level modules with properly calibrated probabilities, so that informed decisions can be made in a well-founded decision-theoretic framework.

To our knowledge, the only method for using graph cuts to produce probabilistic marginals is based on the work of Kohli & Torr (KT) [8]. In this paper, we hope to provide additional insight into the practice of using graph cuts to construct probabilistic models, by framing the method of KT as exact marginal inference in a model that we will elaborate on in later sections. Practically, our goal in this work is to revisit the question of how graph cuts can be used to produce proper uncertainty in random field models. Perhaps surprisingly, we will leave the test-time inference procedure of KT unchanged. develop a new training procedure that directly considers the question of how to set parameters so that the method of KT produces well-calibrated test-time marginal probabilities. We will show that with this new training procedure, graph cuts can be made to produce very good measures of uncertainty. We then show how this same concept enables us to generalize the binary graph cuts model to multi-label data.

We make several contributions:

- We develop theoretical underpinnings for the inference procedure of KT, showing that there is a generative probabilistic model for which their inference procedure produces exact probabilistic marginals.
- We show how to efficiently train this new generative model under the maximum likelihood objective, and develop an algorithm for efficiently computing subgradients using dynamic graph cuts.
- We develop a new model of multilabel data, where exact marginals and subgradients can be computed efficiently using dynamic graph cuts.
- We show empirically that our approach produces better measures of uncertainty than the method of KT and loopy belief propagation-based learning.
- We show our properly calibrated marginal probabilities can be used in a decision theoretic framework to approximately optimize test performance on the intersection-over-union ($\frac{\cap}{\cup}$) loss function, and we show empirically that this improves test performance.

2. Background

Our task is to produce a distribution over a D-dimensional space $\mathcal{Y} = \{1, \ldots, K\}^D$ in which each component takes one of K discrete values. In particular, this distribution should be conditioned upon a feature vector $\boldsymbol{x}$, which takes values in $\mathcal{X}$. This is known as a conditional random field (CRF) model. Our training data are N feature/label pairs, $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$, $\boldsymbol{y}^{(n)} \in \mathcal{Y}$. We will proceed by constructing a model $p(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{w})$ parameterized by weights $\boldsymbol{w}$. As is typical, we will assume that the $\boldsymbol{y}^{(n)}$ are independent of each other, given the $\boldsymbol{x}^{(n)}$ and $\boldsymbol{w}$. The classical formulation of the CRF likelihood function in this setting is to construct an energy function $E(\boldsymbol{y}\,;\,\boldsymbol{x}, \boldsymbol{w})$ and use the Gibbs distribution:

$$p(\boldsymbol{y} \,|\, \boldsymbol{w}, \boldsymbol{x}) = \frac{1}{\mathcal{Z}(\boldsymbol{w}, \boldsymbol{x})} \exp\left\{-E(\boldsymbol{y}\,;\,\boldsymbol{x}, \boldsymbol{w})\right\} \qquad (1)$$

$$\mathcal{Z}(\boldsymbol{x}, \boldsymbol{w}) = \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp\left\{-E(\boldsymbol{y}'\,;\,\boldsymbol{x}, \boldsymbol{w})\right\}. \qquad (2)$$

One natural way to formulate the problem of learning an appropriate $\boldsymbol{w}$ from the data is to maximize the the log likelihood of the training data,

$$L(\boldsymbol{w}; \mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} \log p(\boldsymbol{y}^{(n)} \,|\, \boldsymbol{x}^{(n)}, \boldsymbol{w}). \qquad (3)$$

Optimization of Eq. 3 is often difficult due to the fact that the gradients require computing expectations which are sums over an exponentially large set of states. Various approximation schemes (e.g., [12, 3]) have been developed to attempt to grapple with this difficulty.

Given parameters, we then need to perform inference. Even restricted to the case of graph-structured submodular interactions over binary variables, computing exact probabilistic marginals is intractable due to the difficulty of computing the partition function [7]; however, MAP inference can be performed exactly in low-order polynomial time using the graph cuts algorithm which reduces the problem to the computation of maximum flow in a network [6].

In addition to the solution to the MAP inference problem, we will also make use of quantities known as *min-marginals*. Whereas the value of the MAP solution is $\min_{\hat{\boldsymbol{y}} \in \mathcal{Y}} E(\hat{\boldsymbol{y}}\,;\,\boldsymbol{x}, \boldsymbol{w})$, min-marginals are defined as the value of a constrained minimization problem where a single variable y_d is clamped to take on label k, then all other variables are minimized out:

$$\Phi_d(k) = \min_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = k} E(\hat{\boldsymbol{y}}\,;\,\boldsymbol{x}, \boldsymbol{w}). \qquad (4)$$

This constrained minimization problem can also be solved efficiently using graph cuts, and the set of all min-marginals $\{\Phi_d(k)\}_{d=1:D, k=1:K}$ can be computed in only slightly more time than is required to solve a single graph cuts problem, by using the dynamic graph cuts approach of [8]. Kohli and Torr [8] further suggest that min-marginals can be used to produce a measure of uncertainty q by taking a softmax over the negative min-marginals: $q_d(k) = \frac{\exp\{-\Phi_d(k)\}}{\sum_{k'} \exp\{-\Phi_d(k')\}}$. Given these marginals, they further suggest that a CRF model can be trained by replacing exact marginals needed for the gradient with these approximate marginals. We will evaluate this learning method in the experiments section.

Finally, we will make use of assignments that we will term *argmin-marginals*,

$$\boldsymbol{\eta}_d(k) = \operatorname*{arg\,min}_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = k} E(\hat{\boldsymbol{y}}\,;\,\boldsymbol{x}, \boldsymbol{w}), \qquad (5)$$

which simply replaces the $\min$ in min-marginals with $\arg\min$. These also can be computed efficiently using dynamic graph cuts.

Limitations of Kohli-Torr. In the supplementary material, we discuss the worst-case behavior of KT, showing that even for pairwise graphical models, the KT-estimated marginal can differ from the Gibbs distribution marginal by a factor that has exponential dependence on the number of variables in the model.

3. Our Model

In this paper, we avoid the problem of approximating CRF marginals, and in fact avoid the problem of a complicated partition function altogether. We do this by defining the following generative model:

$$\Phi_d(k;\boldsymbol{x},\boldsymbol{w}) = \min_{\hat{\boldsymbol{y}}\in\mathcal{Y},\hat{y}_d=k} E(\hat{\boldsymbol{y}}\,;\,\boldsymbol{x},\boldsymbol{w})$$

$$p(y_d\!=\!k \,|\, \{\Phi_d(k';\boldsymbol{x},\boldsymbol{w})\}_{k'=1}^K) = \frac{e^{-\Phi_d(k;\boldsymbol{x},\boldsymbol{w})}}{\sum_{k'=1}^K e^{-\Phi_d(k';\boldsymbol{x},\boldsymbol{w})}}.$$

We are here denoting the dth component of $\boldsymbol{y}$ and $\boldsymbol{y}'$ by y_d and y'_d, respectively. We then interpret the min-marginals as providing a fully-factorized distribution on $\boldsymbol{y}$ given $\boldsymbol{x}$.

In contrast to Gibbs-based energy models, this procedure is truly generative: we compute the min-marginals and this gives rise to local distributions over labels. The likelihood for $\boldsymbol{w}$ is then given by

$$p(\{\boldsymbol{y}^{(n)}\}_{n=1}^N \,|\, \boldsymbol{w},\{\boldsymbol{x}^{(n)}\}_{n=1}^N) = \prod_{n=1}^N \prod_{d=1}^D \prod_{k=1}^K q_{ndk}^{\delta(y_d^{(n)},k)}, \qquad (6)$$

where $q_{ndk} = \frac{\exp\left\{-\Phi_d^{(n)}(k;\boldsymbol{x},\boldsymbol{w})\right\}}{\sum_{k'}\exp\left\{-\Phi_d^{(n)}(k';\boldsymbol{x},\boldsymbol{w})\right\}}$, and $\delta(\cdot,\cdot)$ is the Kronecker delta function. This likelihood makes the nature of the model clear: we are parameterizing a large set of multinomial distributions with $\boldsymbol{x}$ and $\boldsymbol{w}$. It simply happens that the parameters of these multinomials are the result of a set of constrained energy minima. Importantly, we can compute q's and thus compute these marginals efficiently when $E(\boldsymbol{y}\,;\,\boldsymbol{x},\boldsymbol{w})$ is a binary submodular energy function, using the approach of Kohli & Torr.

For the binary model we use in much of this paper, we will assume that the weights $\boldsymbol{w}$ parameterize the energy via a sum of weighted unary and pairwise potentials:

$$E(\boldsymbol{y};\boldsymbol{x},\boldsymbol{w}) = \sum_{f\in\mathcal{U}} w_f\psi_f(\boldsymbol{y};\boldsymbol{x}) + \sum_{f\in\mathcal{P}} w_f\psi_f(\boldsymbol{y};\boldsymbol{x}), \qquad (7)$$

where $\mathcal{U}$ and $\mathcal{P}$ are the sets of unary and pairwise features, respectively. The potentials are sums over all local configurations: $\psi_f(\boldsymbol{y};\boldsymbol{x}) = \sum_d \psi_{f,d}(\boldsymbol{y};\boldsymbol{x})$ for $f\in\mathcal{U}$ and $\psi_f(\boldsymbol{y};\boldsymbol{x}) = \sum_{(d,d')}\psi_{f,dd'}(\boldsymbol{y};\boldsymbol{x})$ for $f\in\mathcal{P}$; the local configurations have the form:

$$\psi_{f,d}(\boldsymbol{y};\boldsymbol{x}) = \begin{cases} \alpha_{f,d}(\boldsymbol{x}) & \text{if } y_d = 1 \\ 0 & \text{otherwise} \end{cases} \qquad (8)$$

$$\psi_{f,dd'}(\boldsymbol{y};\boldsymbol{x}) = \begin{cases} \beta_{f,dd'}(\boldsymbol{x}) & \text{if } y_d \neq y_{d'} \\ 0 & \text{otherwise} \end{cases}. \qquad (9)$$

Here, $\alpha_{f,d}(\boldsymbol{x})$ (or $\beta_{f,dd'}(\boldsymbol{x})$) are the result at location d (or edge dd') of running a predefined filter f on input $\boldsymbol{x}$.

4. Maximum Likelihood Learning

As our goal is to produce well-calibrated conditional probabilities for test data, the natural training objective is to maximize the (possibly penalized) likelihood. That is, given a set of observations $\mathcal{D}\!=\!\{\boldsymbol{x}^{(n)},\boldsymbol{y}^{(n)}\}_{n=1}^N$, we wish to find the MLE (or MAP) of the parameters $\boldsymbol{w}$. In this section, we show that subgradients of this objective can be computed efficiently for any model where we have efficient procedures for computing min-marginals.

In reality, images may be of different sizes. To remove the bias that larger images have a larger effect on the learning than smaller images, we rescale likelihoods and instead sum the *average* log likelihood of each instance. Note that if all images are of the same size, optimizing this objective is equivalent to optimizing the earlier objective Eq. 6. The objective for the nth data instance can then be written as

$$L^{(n)}(\boldsymbol{w}) = -\frac{1}{D^{(n)}}\sum_{d=1}^{D^{(n)}}\Bigg[\Phi_d(y_d^{(n)};\boldsymbol{w},\boldsymbol{x}^{(n)}) + \log\sum_{k=1}^K \exp\left\{-\Phi_d(k;\boldsymbol{w},\boldsymbol{x}^{(n)})\right\}\Bigg]. \qquad (10)$$

We are interested in the partial derivative with respect to one parameter, say w_f. Dropping superscripts n to reduce notational clutter,

$$\frac{\partial L(\boldsymbol{w})}{\partial w_f} = \frac{1}{D}\sum_{d=1}^D\sum_{k=1}^K \frac{\partial L(\boldsymbol{w})}{\partial\Phi_d(k;\boldsymbol{w},\boldsymbol{x})}\frac{\partial\Phi_d(k;\boldsymbol{w},\boldsymbol{x})}{\partial w_f}. \qquad (11)$$

The first term is a standard softmax derivative:

$$\frac{\partial L(\boldsymbol{w})}{\partial\Phi_d(k)} = -\delta(y_d,k) + \frac{\exp\{-\Phi_d(k;\boldsymbol{w},\boldsymbol{x})\}}{\sum_{k'}\exp\{-\Phi_d(k';\boldsymbol{w},\boldsymbol{x})\}}. \qquad (12)$$

To compute the second term, first expand the definition of $\Phi_d(k;\boldsymbol{w},\boldsymbol{x})$, then compute a subgradient:

$$\frac{\partial\Phi_d(k;\boldsymbol{w},\boldsymbol{x})}{\partial w_f} = \frac{\partial}{\partial w_f}\min_{\hat{\boldsymbol{y}}\in\mathcal{Y},\hat{y}_d=k'}\sum_f w_f\psi_f(\hat{\boldsymbol{y}};\boldsymbol{x}) = \psi_f(\boldsymbol{\eta}_{dk};\boldsymbol{x}), \qquad (13)$$

where recall $\boldsymbol{\eta}_{dk} = \arg\min_{\hat{\boldsymbol{y}}\in\mathcal{Y},\hat{y}_d=k} E(\hat{\boldsymbol{y}}\,;\,\boldsymbol{x},\theta)$ is the argmin-marginal for $y_d = k$. The total subgradient for one instance is then

$$\frac{\partial L(\boldsymbol{w})}{\partial w_f} = \frac{1}{D}\sum_{d=1}^D\sum_{k=1}^K \psi_f(\boldsymbol{\eta}_{dk};\boldsymbol{x})\,[q_{ndk}-\delta(y_d,k)]. \qquad (14)$$

Using these gradients we can train the model to optimize the likelihood of the training data.

5. Faster Computation of Subgradients

The subgradients in Eq. 14 naively take $O(D^2)$ time to compute, which can be expensive for large images. In this section, we show how to significantly reduce this time by (a) leveraging the locality of changes within the dynamic graph cuts procedure used to compute min-marginals; (b) reordering the computation of min-marginals; and (c) distributing computation across many CPUs. The result of (a) is that computation of gradients is only a constant factor slower than computing min-marginals; (b) speeds up the computation of min-marginals and thus subgradients; and (c) allows us to easily scale to large data sets, assuming we have access to a large cluster of machines.

(a) Locality of Changes in Argmin Marginals. The maxflow algorithm of [1] caches search trees from iteration to iteration. The only nodes that can change are ones that are “orphaned” (that is, their connection to the root of the search tree is severed) after an edge capacity modification or subsequent path augmentations. This list of potentially changed nodes can be stored during the graph cuts procedure (this option is available in the code of Kolmogorov [1]), and it is typically much smaller than D. So in the inner loop, we look only at potentially changed nodes.

This modification makes the subgradient computations equivalent in computational cost to computing min-marginals, up to a constant factor, because the subgradient computation only considers nodes that are processed in the min-marginal computation. In Section 7, we compare the time taken using our method to the time taken using only min-marginals and confirm that this holds empirically.

(b) Ordering Min-marginal Computations. Computing min-marginals requires solving $D+1$ graph cuts problems. The cost is greatly reduced by using dynamic graph cuts, but we have found experimentally that the order of problems can make a large difference in the time it takes to compute min-marginals. The strategy we use is as follows: first, compute the MAP; next, compute min-marginals for variables that take on value 0 in the MAP assignment, iterating over the variables in scanline ordering; finally, compute min-marginals for variables that take on value 1 in the MAP assignment, iterating over the variables in scanline ordering. The intuition for this order is dynamic graph cuts is more efficient when the initial solution is closer to the final solution. If after clamping a variable $y_d = 0$, the neighboring variable $y_{d'}$ is also clamped to 0, solutions will tend to be more similar than if $y_d = 1$. This effect tends to increase as pairwise potentials become stronger.

(c) Distributed Computation. Gradients can be computed for each image in parallel, enabling distribution of the learning algorithm over multiple cores. In our implementation, we used C++ to build a distributed learning system in which one master process communicates with the workers via RPC or MPI. The master sends the workers a current setting of weights, and each worker returns a vector of gradients. The master accumulates the gradients, updates weights, then sends out a new request. This process repeats until termination. This parallelization resulted in an almost linear speedup with the number of cores.

6. Tractable Multilabel Model

In the multilabel setting, MAP inference becomes NP-hard in most cases [2], so we cannot compute exact min-marginals $\Phi_d(k; \boldsymbol{x}, \theta)$; thus, it appears that the model presented above cannot be applied. Notice, however, that there is no requirement in our generative model that the $\Phi_d(k; \boldsymbol{x}, \theta)$ values correspond to exact min-marginals. We require only that they be a deterministic function of parameters, that they be efficiently computable, and that we can compute subgradients of them with respect to model parameters. In this section, we replace the intractable multilabel min-marginal calculations with a tractable surrogate.

For multilabel models, as is typical, we let there be a separate set of weights for each feature f and class k, defining e.g., the unary potential for pixel d taking on label k as $\theta_d(k) = \sum_f w_f^k \psi_{f,d}(k; \boldsymbol{x})$. In this section, to represent a multilabel assignment for pixel d, we will use K binary variables, $y_{d1}, \dots, y_{dK}$. We then define separate energy functions for each $k \in \{1, \dots, K\}$:

$$E^k(\boldsymbol{y}; \boldsymbol{x}, \theta) = \sum_{d \in \mathcal{V}} \theta_d^k(y_{dk}) + \sum_{dd' \in \mathcal{E}} \theta_{dd'}^k(y_{dk}, y_{d'k}), \quad (15)$$

where $\theta_d^k(0) = 0$, $\theta_d^k(1) = \theta_d(k)$, and $\theta_{dd'}^k(y_{dk}, y_{d'k})$ are pairwise potentials with different parameters per k.

We can then define separate min-marginals $\Phi_d^k(y_{dk}) = \min_{\hat{y}_d = y_{dk}} E^k(\hat{\boldsymbol{y}}; x, \theta)$. These can be computed exactly using a graph cuts min-marginals computation for each k. Finally, we define multilabel surrogate min-marginals to be $\tilde{\Phi}_d(k) = \Phi_d^k(1) - \Phi_d^k(0)$, then let $q_{dk} = \frac{\exp\{-\tilde{\Phi}_d(k)\}}{\sum_{\hat{k}} \exp\{-\tilde{\Phi}_d(\hat{k})\}}$ be the multilabel probability of pixel d taking label k.

These surrogate min-marginals then have the properties that we desire: they are deterministic, efficiently computable, and we can (sub)differentiate through them. They do not correspond to min-marginals for a CRF model, but we can think of them as coming from a some other generative process where exact maximum likelihood learning and marginal inference are tractable via graph cuts.

We have seen in the previous section how to derive $\frac{\partial \Phi_d^k}{\partial w_f}$. To derive subgradients for the multilabel model, we simply need to observe that $\frac{\partial \tilde{\Phi}_d(k)}{\partial w_f} = \frac{\partial \Phi_d^k(1)}{\partial w_f} - \frac{\partial \Phi_d^k(0)}{\partial w_f}$. Focusing on a single instance,

$$\frac{\partial L(\boldsymbol{w})}{\partial w_f^k} = \frac{1}{D} \sum_{d,k'} \frac{\partial L}{\partial \tilde{\Phi}_d(k')} \left(\frac{\partial \Phi_d^{k'}(1)}{\partial w_f^k} - \frac{\partial \Phi_d^{k'}(0)}{\partial w_f^k} \right). \quad (16)$$

The first term is a standard softmax derivative, just as before. For both unary and pairwise features f,

$$\frac{\partial \Phi_d^{k'}(y_{dk'})}{\partial w_f^k} = \mathbf{1}_{\{k=k'\}} \psi_f(\boldsymbol{\eta}_d^{k'}(y_{dk'}); \boldsymbol{x}), \tag{17}$$

where $\boldsymbol{\eta}_d^k(y_{dk}) = \arg\min_{\hat{y}_d = y_{dk}} E^k(\hat{\boldsymbol{y}}; x, \theta)$. These subgradients can also be computed efficiently using the methods described in Section 5.

7. Experiments

Experimentally, we apply our models to image segmentation tasks and investigate three main questions. The first is how well our method can optimize the maximum likelihood objective. We compare against the learning method suggested by Kohli & Torr (KT), and against a logistic regression baseline. Second, we look at the generalization capabilities of our models — both the binary and multilabel variants. Our main evaluation measure is the probability assigned to held-out test examples, but we also look at hard predictive performance, measured in terms of test accuracy and area under the ROC curve. Third, we investigate the suitability of the marginals for driving decision theoretic predictions in terms of expected loss.

We use 84 unary and 4 pairwise features. The unary features are simple color-based and texture-based filters, run on patches surrounding the pixel. One pairwise feature is uniformly set to 1, while the others are based on thresholded responses of the pb boundary detector [10]. We emphasize that these features include only low level cues.

For our experiments, we use a subset of the PASCAL VOC Image Segmentation data. We build binary datasets by considering only images containing a given object class (e.g., airplane), then the task is to label the given object pixels as “figure” and all other pixels as “ground”. We build multilabel datasets by taking a subset of classes and only considering images that have at least one of the selected classes present. Images are scaled so the minimum dimension is 100 pixels. We focused on Aeroplane, Car, Cow, and Dog classes but expect results to be representative of the case where unary information is fairly weak, due to the simplicity of our input features. We believe to be a common and important case to consider for low-level vision systems.

7.1. Evaluation of Binary Model Optimization

Recall that the test-time procedures for our method and KT are identical. Consequently, we can compare the effectiveness of training the model described in Section 3 using softmaxed negative min-marginals as approximate gradients (as in [8]) versus using exact subgradients (our method). We also consider a baseline with no pairwise potentials. The likelihood evaluations of these models are focused on the case where the goal at test time is to produce a pixel-wise measure of uncertainty, as would be appropriate in e.g. interactive image segmentation, multiscale segmentation, and in the decision theoretic prediction setting that appears in Section 7.2.

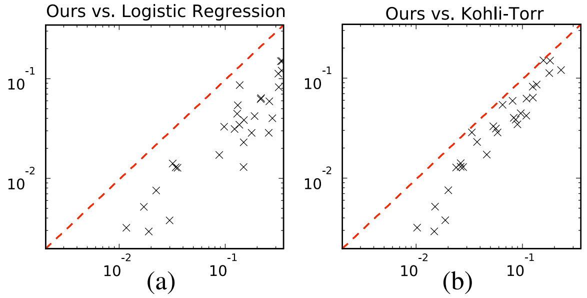


Figure 1. Comparison of training negative log likelihoods achieved by our method (y-axis) versus (a) logistic regression, and (b) the KT method (x-axis). There is one marker for each of 30 images, which were optimized independently. In all cases, we achieve better training likelihoods than the alternative methods.

Single Image Datasets. For the first experiment, we considered 30 data sets, each with a single aeroplane instance. We optimized the logistic regression model to convergence using gradient ascent, then we initialized the other two methods with the result, initially setting all pairwise weights to zero. We then ran gradient-based optimization using the (sub)gradients computed by the two methods and recorded the best objective achieved. For KT, we followed [8] and used a fixed step size that was tuned by hand but left fixed across experiments. For our method, we used a dynamic step size decay schedule, which we found in practice to outperform various static decay schedules: we maintain a quantity $f_{best}^{(t)} = \min_{t' \in \{1,\dots,t\}} f(\theta_{t'})$, where f is the negative average log likelihood objective function. We then use $\lambda \cdot f_{best}^{(t)}$ as an estimate of the optimal value f^* at iteration t and perform a Polyak-like update, setting step size $\alpha_t = \frac{f(\theta_t) - \lambda f_{best}^{(t)}}{||g||^2}$, where g is the subgradient. We chose $\lambda = .95$ and left it fixed across experiments. (We also experimented with dynamic step size decay schedules for the KT gradients, but we could not get them to outperform the fixed update schedule.) Results are shown in Fig. 1. While KT always produces better likelihoods than a unary only (logistic regression) model, its gradients are not directly optimizing this quantity (and indeed, it is unclear that there is any quantity being exactly optimized with the KT approach). When the correct gradients are used (our method), we achieve much better training likelihoods.

Full Datasets. Next, we focused on the comparison to KT and experimented with larger data sets. For each class, we constructed a training set with 48 images, and parallelized the optimizations over 17 CPUs (1 master, 16 workers). In Fig. 2, we show the best training objective achieved as a function of wall-clock time. An iteration of KT is faster than an iteration of our method (due to the fact that we need to compute argmin-marginals in addition to min-

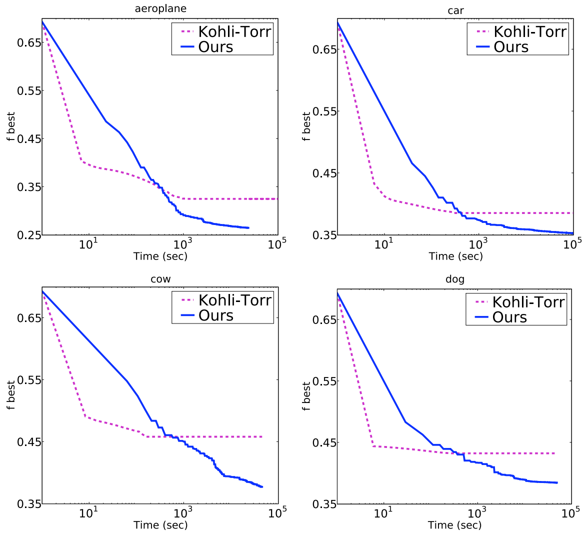


Figure 2. f_{best} versus time for learning on full training sets.

marginals), but within 1000 seconds, our method overtakes KT, and then always leads to better training likelihoods.

7.2. Evaluation of Binary Model

We then compare the results of our optimization to that of KT in terms of performance as a model of segmentation data. We constructed a test set with the remaining images (roughly 50 per class) not used for training. We observed that KT tended towards a set of weights that were different from the weights that achieved the best performance under the maximum likelihood objective. To give a better representation of the behavior, we report results for the model at two points: first, **KT-f_{best}** takes the weights that achieve the best training likelihood objective. Second, we let the model run for longer and took a set of weights from the point that it seemed to converge to. We call this **KT-final**.

Train and Test Performance. In the first set of evaluations, we report training and test performance according to three measures: average pixel likelihood, 0-1 pixel accuracy, and area under the ROC curve (AUC). Our approach is consistently best on the likelihood and AUC measures, which are the ones where a good measure of uncertainty is required, and it is competitive on pixel accuracy in all cases. In the Car data, all methods experienced some overfitting, but otherwise training performance was indicative of test performance, showing that better optimization of the maximum likelihood objective led to models with better test performance. Quantitative results are shown in Fig. 3, and illustrative qualitative results are shown in Fig. 4.

Decision Theoretic Predictions for $\frac{\cap}{\cup}$ Score. Given properly calibrated probabilities, we can make predictions that seek to maximize expected score on the test set. Here, we take this approach and seek to optimize the intersection-over-union ($\frac{\cap}{\cup}$) score that is commonly used to evaluate image segmentations. Given true labeling $\boldsymbol{y}^*$, the score is defined as $\Delta(\boldsymbol{y}, \boldsymbol{y}^*) = \frac{1}{K}\sum_{k=1}^{K} \frac{\sum_d \mathbf{1}_{\{y_d^*=k \wedge y_d=k\}}}{\sum_d \mathbf{1}_{\{y_d^*=k \vee y_d=k\}}}$. In the case of a binary model, $K=2$ and classes are foreground and background. Given our predictive distribution $Q(\boldsymbol{y}) = \prod_{d=1}^{D}\prod_{k=1}^{K} q_{dk}^{\delta(y_d,k)}$, the expected score for making prediction $\boldsymbol{y}$ is $e(\boldsymbol{y}) = \sum_{\boldsymbol{y}'} Q(\boldsymbol{y}')\Delta(\boldsymbol{y}, \boldsymbol{y}')$. Since Δ does not decompose, even evaluating $e(\boldsymbol{y})$ requires a sum over exponentially many joint configurations. Instead, we define a smoothed surrogate expected score that is tractable to evaluate given prediction $\boldsymbol{y}$: $\tilde{e}(\boldsymbol{y}) = \frac{1}{K}\sum_{k=1}^{K} \frac{\mathbb{E}_{Q(\boldsymbol{y}')}[\sum_d \mathbf{1}_{\{y'_d=k \wedge y_d=k\}}]}{\mathbb{E}_{Q(\boldsymbol{y}')}[\sum_d \mathbf{1}_{\{y'_d=k \vee y_d=k\}}]}$. Our strategy will be to

		Log Lik	Accuracy	AUC
Aero	KT-final	-.35 (-.29)	87.3 (89.7)	.85 (.87)
	KT-f_{best}	-.32 (-.28)	88.1 (89.9)	.85 (.87)
	Ours	**-.26** **(-.24)**	**88.9** **(90.4)**	**.90** **(.90)**
Car	KT-final	-.48 (-.65)	84.1 (78.7)	.69 **(.65)**
	KT-f_{best}	-.39 **(-.51)**	**86.2** (80.0)	.66 (.62)
	Ours	**-.35** **(-.51)**	86.1 **(81.4)**	**.76** **(.65)**
Cow	KT-final	-.54 (-.64)	79.7 (75.9)	.76 (.77)
	KT-f_{best}	-.47 (-.52)	80.9 (76.7)	.66 (.65)
	Ours	**-.38** **(-.41)**	**82.5** **(79.9)**	**.84** **(.82)**
Dog	KT-final	-.52 (-.45)	81.8 (84.7)	.64 (.66)
	KT-f_{best}	-.43 (-.38)	**84.1** (86.7)	.62 (.66)
	Ours	**-.38** **(-.34)**	84.0 **(86.8)**	**.76** **(.79)**

Figure 3. Results for binary models. Format is "Train (Test)".

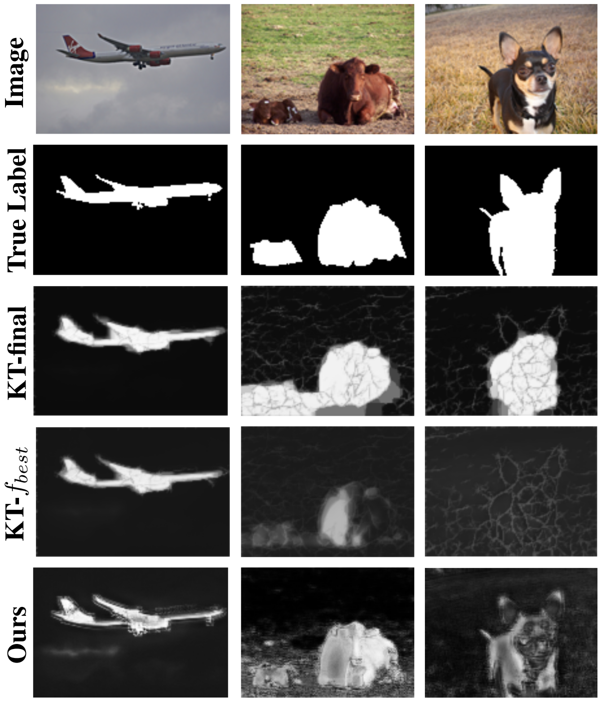


Figure 4. Estimated marginal probabilities for test examples. On many examples (e.g. left), the three methods behave similarly. When they behave differently (middle and right), KT-final often becomes overconfident; KT-f_{best} is often under-confident; and our method is more able to produce well-calibrated probabilities.

		$\cap/\cup$ **Before**	$\cap/\cup$ **After**	**Change**
Aero	KT-final	63.4	63.9	.5
	KT-f_{best}	60.7	61.8	1.1
	Ours	**64.1**	**66.6**	**2.5**
Car	KT-final	**43.5**	44.2	.7
	KT-f_{best}	40.7	43.3	2.6
	Ours	41.8	**45.9**	**4.1**
Cow	KT-final	**51.2**	51.7	.5
	KT-f_{best}	40.7	47.1	**6.4**
	Ours	46.5	**52.8**	6.3
Dog	KT-final	**52.1**	52.0	-.1
	KT-f_{best}	44.5	47.2	2.7
	Ours	48.9	**55.3**	**6.4**

Figure 5. Test results for maximizing surrogate expected $\frac{\cap}{\cup}$ score. "Before" corresponds to predicting the mode of Q; "After" is the prediction from our expected score maximization routine.

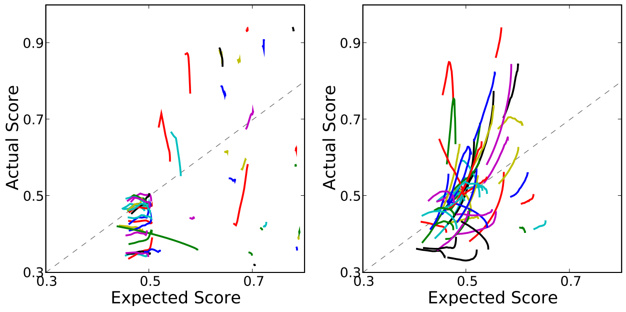
initialize prediction $\boldsymbol{y}$ at the mode of Q, then to greedily hill climb in terms of $\tilde{e}(\boldsymbol{y})$ until we reach a local maximum of expected score. At each step, we iterate through classes k, proposing to flip pixel d to label k, where d is the pixel that has largest probability q_{dk} amongst pixels not currently labeled k. When we cycle through all k but do not make a flip, we terminate. This yields our prediction, $\boldsymbol{y}$, which we will evaluate under $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$. Quantitative results for this approach are shown in Fig. 5. Interestingly, even though KT-final gives higher scores for the initial mode prediction, our method surpasses it in all cases after running the expected score optimization. Because KT-final does not produce well-calibrated probabilities, the expected loss optimization either provides little win or hurts predictions.

In Fig. 6, we illustrate the trajectories that the expected score optimizer takes as it performs the local ascent. Each line is for a different image, and the left-most endpoint of the line corresponds to the initialization of the optimizer. As the line moves right, the expected score increases, and ideally the true score will also increase, which would correspond to the line moving upwards. In Fig. 7, we show the change in predictions from before and after running the optimizer for three images, under our predictive distributions.

Statistical Significance. For all of the experiments in this section, we ran a bootstrap experiment, where we resampled instances with replacement, and computed the mean of each evaluation measure on each resampled set of instances. We repeated the resampling procedure 1000 times and computed the standard deviations across the resampled datasets. Tables with these error bars appear in the Supplementary Material.

7.3. Evaluation of Multilabel Model

Finally, we ran experiments on the multilabel model, and compared it to learning a CRF with loopy belief propagation (LBP) for approximate inference. We used the publicly available libDAI implementation of LPB [11], setting damping to .3, and using a maximum of 100 iterations. We constructed a dataset of 5 classes (the four from previously plus background), and chose 80 images evenly from the 4 foreground classes. We similarly constructed a test set of a separate 80 images. To optimize, we parallelized computation across 41 CPUs (1 master, 40 slaves), and let each algorithm run for 12 hours (nearly 500 hours of CPU time). The models produced similar test performance — LBP gave an average test $\frac{\cap}{\cup}$ score of 17.2, while ours produced a score of 17.7. After expected score optimization, LBP performance increased to 17.3, while ours increased to 20.0. However, the most striking difference between the approaches was the



(a) **KT-final** (b) **Ours**

Figure 6. Trajectories as the local optimizer moves from mode prediction (left-most point of line segment) to the prediction that locally maximizes the surrogate expected score (right-most point of line segment). The surrogate expected score is on the x-axis, and the true score (which uses the ground truth to compute) is on the y-axis. (a) KT-final on dog test data. (b) Ours on dog test data.

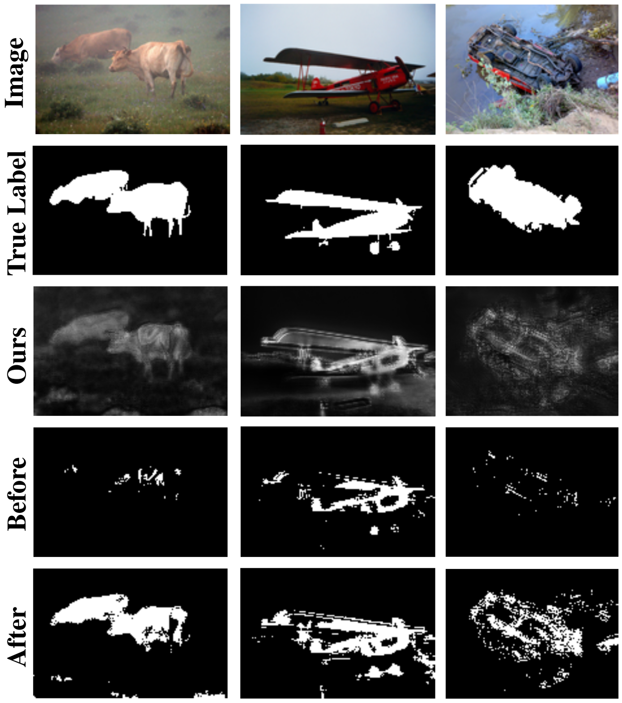


Figure 7. Expected loss optimization results on test images.

	Time (sec)	% Not Conv.
Loopy BP (first iter.)	11.3 ± 1.7	0%
Ours (first iter.)	**0.14 ± .02**	–
Loopy BP (last iter.)	59.1 ± 15.1	30%
Ours (last iter.)	**0.16 ± .02**	–

Figure 8. Time taken for a single inference call on multilabel models, reported early and late in learning. As pairwise potentials get stronger, loopy BP gets slower and less reliable; the graph cuts inference is uniformly reliable and two orders of magnitude faster.

speed and reliability of the inference routines. While LBP was consistently more than two orders of magnitude slower, performance got even worse as learning progressed, and we later had problems with non-convergence. Conversely, the graph cuts based inference was uniformly fast and reliable. Quantitative results are shown in Fig. 8.

8. Discussion and Related Work

Our approach is a deviation from the standard strategy of defining an intractable model, then devising efficient but approximate inference routines. Instead, we are taking an efficient inference routine and treating it as a model. Specifically, we asked, "for what model is the method of [8] an exact marginal inference routine?" The answer is the model that we presented in Section 3. The power of this approach is that we can efficiently compute exact gradients of this model under the maximum likelihood objective (Section 4), so we are directly training the graph cuts inference to produce well calibrated marginal probabilities at test time. In Section 6, we show how to extend the ideas to multilabel problems where MAP inference is NP-hard. The key idea is to build a model composed of tractable subcomponent modules, which are as expressive as possible while still admitting efficient exact inference. We showed experimentally that this approach gives strong empirical performance.

If we look at a high level and consider works that define models around efficient computational procedures, there is some related work. [13] defines a generative probabilistic models that includes a discrete optimization procedure as the final step. [4] defines probability models around a fixed number of iterations of belief propagation. None of these is a min-marginal computation, and thus the specifics are quite different, but the general spirits are similar.

At a broader level, we are addressing a *low level* vision problem in this work. While low level vision has received considerable attention in computer vision, there has not been a strong emphasis on producing properly calibrated probabilistic outputs. Our approach maintains the computational efficiency of previous surrogate measures of uncertainty, but it does so within a proper probabilistic framework. We believe this direction to be of importance going forward when building large probabilistic vision systems. There are also direct applications to multiscale image labeling, interactive image segmentation, and active learning.

Finally, our formulation is quite general, and applies to any model that can be assembled from components where min-marginals can be computed efficiently. Our multilabel model is one example of how to assemble graph cuts components. A similar approach also may be attractive in other structured output domains, such as those with bipartite matching and shortest path structures, where min-marginals can be computed efficiently [5] but where exact marginal inference in the standard CRF formulation is NP-hard.

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI*, 26:1124–1137, 2004. 1, 4

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV*, pages 377–384, 1999. 1, 4

[3] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *International Conference on Artificial Intelligence and Statistics*, 2005. 2

[4] J. Domke. Parameter learning with truncated message-passing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 8

[5] J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief propagation. In *NIPS*, 2007. 1, 8

[6] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51:271279, 1989. 2

[7] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087–1116, 1993. 1, 2

[8] P. Kohli and P. H. S. Torr. Measuring uncertainty in graph cut solutions. *Computer Vision and Image Understanding*, 112(1):30–38, 2008. 1, 2, 5, 8, 9

[9] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts – a review. *PAMI*, 29(7):1274–1279, 2007. 1

[10] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture. In *NIPS*, 2002. 5

[11] J. M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *JMLR*, 11:2169–2173, Aug. 2010. 7

[12] I. Murray and Z. Ghahramani. Bayesian learning in undirected graphical models: Approximate MCMC algorithms. In *Unc. in Art. Intel.*, pages 392–399, 2004. 2

[13] G. Papandreou and A. Yuille. Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011. 8

Revisiting Uncertainty in Graph Cut Solutions — Supplementary Material

Analysis of Kohli-Torr as Approximate Gibbs Distribution Inference

Here we treat the min-marginal uncertainty of Kohli & Torr (KT) [8] as a procedure for computing approximate marginal probabilities under the Gibbs distribution that is standard in conditional random field (CRF) models. In the very simple case where there are no interactions between variables in the model (i.e., a "unary only" model), the approximate KT marginals match the true marginals. The main result here is that even after adding just submodular pairwise interactions to the model, the marginals estimated by KT can be off by a factor that is exponentially large in the number of variables in the network.

The crux of the problem (which will be made more precise below) is fairly simple: if we approximate the sum of probabilities over a subset of configurations by instead maximizing over the probabilities of states in the subset, how bad can the approximation be? The analysis simply considers how to parameterize models such that the summation required to compute marginals gives as different value as possible from the maximizations used by KT.

We are concerned with this question, because when learning with the KT procedure, approximate gradients are calculated using these approximate marginals. This analysis adds some theoretical explanation for the empirical behavior of the KT learning procedure in our experiments.

We will consider models over a set of binary variables $\boldsymbol{y} \in \mathcal{Y} = \{0,1\}^D$. We assume we have an energy function $E(\boldsymbol{y})$ (for simplicity dropping the dependence on input $\boldsymbol{x}$ and the explicit dependence on parameters $\boldsymbol{w}$), then under the Gibbs distribution, the probability of a configuration $\boldsymbol{y}$ is

$$p'(\boldsymbol{y}) = \exp\{-E(\boldsymbol{y})\} \tag{18}$$

$$\mathcal{Z} = \sum_{\boldsymbol{y}' \in \mathcal{Y}} p'(\boldsymbol{y}') \tag{19}$$

$$p(\boldsymbol{y}) = \frac{1}{\mathcal{Z}} p'(\boldsymbol{y}). \tag{20}$$

The marginal probability for y_d under the Gibbs distribution is

$$p(y_d) = \sum_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y_d} p(\hat{\boldsymbol{y}}) \tag{21}$$

$$= \frac{\sum_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y_d} \exp\{-E(\hat{\boldsymbol{y}})\}}{\sum_{y'_d \in \{0,1\}} \sum_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y'_d} \exp\{-E(\boldsymbol{y}')\}}. \tag{22}$$

That is, we are summing over the probability of all configurations $\boldsymbol{y}'$ that are consistent with the given setting of y_d.

The KT approximation is to define

$$\tilde{p}(y_d) = \frac{\exp\{-\min_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y_d} E(\hat{\boldsymbol{y}})\}}{\sum_{y'_d \in \{0,1\}} \exp\{-\min_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y'_d} E(\hat{\boldsymbol{y}})\}} \tag{23}$$

$$= \frac{\max_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y_d} \exp\{-E(\hat{\boldsymbol{y}})\}}{\sum_{y'_d \in \{0,1\}} \max_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y'_d} \exp\{-E(\hat{\boldsymbol{y}})\}}. \tag{24}$$

Comparing Eq. 22 to Eq. 24, we see that they are identical, except the maximizations in Eq. 24 are replaced by sums to get the expression Eq. 22 for the true marginals. For convenience, let $\tilde{p}'$ be the unnormalized value of the maximization:

$$\tilde{p}'(y_d) = \max_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y_d} \exp\{-E(\hat{\boldsymbol{y}})\}. \tag{25}$$

and similarly

$$p'(y_d) = \sum_{\hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = y_d} \exp\{-E(\hat{\boldsymbol{y}})\}. \tag{26}$$

The question we ask in the following is, "how large can we make the approximation factor $f = \tilde{p}(y_d)/p(y_d)$, for some d and y_d?" In other words, in the worst case, how bad can the approximation of KT be?

8.1. General Models

We begin with the simplest case, where there are no restrictions on the parameterization of the energy function. In this case, we have the freedom to assign an independent probability value to each of the 2^D joint configurations $\boldsymbol{y}$, so long as we restrict the probabilities to be non-negative, and the sum to be equal to 1.

Without loss of generality, suppose that the KT approximate marginal is $\tilde{p}(y_d = 1) = \lambda$, and thus $\tilde{p}(y_d = 0) = 1 - \lambda$, for $\lambda \in [0,1]$. This implies that

$$\frac{\tilde{p}'(y_d = 1)}{\tilde{p}'(y_d = 0) + \tilde{p}'(y_d = 1)} = \lambda. \tag{27}$$

Again without loss of generality, we can further assume that $\tilde{p}'(y_d = 0) + \tilde{p}'(y_d = 1) = 1$, so

$$\tilde{p}'(y_d = 1) = \lambda, \text{ and} \tag{28}$$

$$\tilde{p}'(y_d = 0) = 1 - \lambda. \tag{29}$$

Let S_0 be the set of states where $y_d = 0$:

$$S_0 = \{\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = 0\} \tag{30}$$

and define S_1 similarly:

$$S_1 = \{\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{y}} \in \mathcal{Y}, \hat{y}_d = 1\} \tag{31}$$

So far, we know that

$$\max_{\hat{\boldsymbol{y}} \in S_1} p'(\hat{\boldsymbol{y}}) = \lambda \tag{32}$$

$$\max_{\hat{\boldsymbol{y}} \in S_0} p'(\hat{\boldsymbol{y}}) = 1 - \lambda. \tag{33}$$

The size of these sets are $|S_0| = |S_1| = 2^{D-1}$, so the largest that $p'(y_d = 1)$ could be is $\lambda \cdot 2^{D-1}$. The proof of this is straightforward: suppose that $p'(y_d = 1) > \lambda \cdot 2^{D-1}$. Then there must be some configuration $\boldsymbol{y} \in S_1$ such that $p'(\boldsymbol{y}) > \lambda$. But this contradicts Eq. 32. Finally, $p'(y_d = 1) = \lambda \cdot 2^{D-1}$ is achievable by assigning $p'(\boldsymbol{y}) = \lambda$ for all $\boldsymbol{y} \in S_1$.

The smallest that $p'(y_d = 0)$ could be is $1 - \lambda$, because probabilities are non-negative, and at least one configuration $\boldsymbol{y} \in S_0$ must have $p'(\boldsymbol{y}) = 1 - \lambda$ to be consistent with Eq. 33.

Since S_0 and S_1 are disjoint, the minimum achievable value of $p(y_d = 0)$ consistent with Eq. 32 and Eq. 33 comes from simultaneously maximizing $p'(y_d = 1)$ and minimizing $p'(y_d = 0)$. This yields

$$p(y_d = 0) = \frac{p'(y_d = 0)}{p'(y_d = 0) + p'(y_d = 1)} \tag{34}$$

$$= \frac{1 - \lambda}{1 - \lambda + \lambda \cdot 2^{D-1}}. \tag{35}$$

The approximation factor is then:

$$\frac{\tilde{p}(y_d = 0)}{p(y_d = 0)} = 1 - \lambda + \lambda \cdot 2^{D-1}, \tag{36}$$

which is exponentially large in D for constant values of λ (say $\lambda = .5$).

8.2. Pairwise Submodular Models

We now consider the case where we restrict the parameterization, so that we cannot choose arbitrary values for the probability assigned to a joint assignment $\boldsymbol{y}$. Instead, we require that the model be a pairwise graphical model with only submodular interactions. In this section, we show that there are such models where the same basic strategy as before is applicable, and thus the approximation factor is still exponentially bad.

Consider a pairwise model where there is a single root, y_r, which is shares an edge with all other variables. There are no other edges, so this is also a tree-structured model. Let $\boldsymbol{\theta}$ be the set of potentials, and define the unnormalized probability as a typical graphical model, where the only unary potential is on the root:

$$p'(\boldsymbol{y}) = \theta_r(y_r) \prod_d \theta_{rd}(y_r, y_d). \tag{37}$$

Let $\theta_r(1) = \lambda$, $\theta_r(0) = 1 - \lambda$ for $\lambda \in [0, 1]$, and parameterize pairwise interactions for all d as follows:

$$\theta_{rd}(y_r = 0, y_d = 0) = 1 \tag{38}$$

$$\theta_{rd}(y_r = 0, y_d = 1) = 0 \tag{39}$$

$$\theta_{rd}(y_r = 1, y_d = 0) = 1 \tag{40}$$

$$\theta_{rd}(y_r = 1, y_d = 1) = 1. \tag{41}$$

This is clearly a submodular interaction. Now, if y_r is constrained to be 1, there are 2^{D-1} joint configurations with support, each getting an unnormalized probability of λ. If y_r is constrained to 0, then there is only a single joint configuration with support (the all 0's assignment), and it gets unnormalized probability of $1-\lambda$. This model is then equivalent to the model constructed in the previous section, and thus the approximation factor is also exponentially bad.

Bootstrap Analysis Results

See Fig. 9 and Fig. 10.

		Log Lik	Accuracy	AUC
Aero	KT-final	-.35±.005 (-.29±.004)	87.3±.2 (89.7±.1)	.85±.018 (.87±.020)
	KT-f_{best}	-.32±.007 (-.28±.005)	88.1±.3 (89.9±.2)	.85±.018 (.87±.019)
	Ours	**-.26±.007** (**-.24±.005**)	**88.9±.3** (**90.4±.2**)	**.90±.014** (**.90±.014**)
Car	KT-final	-.48±.008 (-.65±.008)	84.1±.3 (78.7±.3)	.69±.036 (**.65±.022**)
	KT-f_{best}	-.39±.007 (**-.51±.006**)	**86.2±.3** (80.0±.3)	.66±.029 (.62±.023)
	Ours	**-.35±.009** (**-.51±.006**)	86.1±.5 (**81.4±.3**)	**.76±.035 (.65±.032)**
Cow	KT-final	-.54±.009 (-.64±019)	79.7±.3 (75.9±.8)	.76±.040 (.77±.038)
	KT-f_{best}	-.47±.004 (-.52±.008)	80.9±.3 (76.7±.5)	.66±.040 (.65±.038)
	Ours	**-.38±.005** (**-.41±.006**)	**82.5±.4** (**79.9±.4**)	**.84±.023** (**.82±.034**)
Dog	KT-final	-.52±.009 (-.45±.004)	81.8±.4 (84.7±.1)	.64±.029 (.66±.027)
	KT-f_{best}	-.43±.004 (-.38±.003)	**84.1±.2** (86.7±.1)	.62±.023 (.66±.026)
	Ours	**-.38±.004** (**-.34±.004**)	84.0±.2 (**86.8±.2**)	**.76±.025** (**.79±.028**)

Figure 9. Results for binary models. Format is "Train (Test)". ± indicates the standard deviation of the result under bootstrap resamplings.

		$\cap/\cup$ **Before**	$\cap/\cup$ **After**	**Change**
Aero	KT-final	63.4±2.2	63.9±2.1	.5±.3
	KT-f_{best}	60.7±2.1	61.8±1.9	1.1±.4
	Ours	**64.1±2.2**	**66.6±2.0**	**2.5±.7**
Car	KT-final	**43.5±1.4**	44.2±1.2	.7±.3
	KT-f_{best}	40.7±1.0	43.3±.9	2.6±.5
	Ours	41.8±1.0	**45.9±1.0**	**4.1±.8**
Cow	KT-final	**51.2±3.7**	51.7±3.6	.5±.5
	KT-f_{best}	40.7±1.8	47.1±2.7	**6.4±2.0**
	Ours	46.5±1.9	**52.8±2.8**	6.3±1.8
Dog	KT-final	**52.1±2.6**	52.0±2.5	-.1±.5
	KT-f_{best}	44.5±1.1	47.2±1.2	2.7±.6
	Ours	48.9±1.4	**55.3±1.9**	**6.4±1.3**

Figure 10. Test results for maximizing surrogate expected $\frac{\cap}{\cup}$ score. "Before" corresponds to predicting the mode of Q; "After" is the prediction from our expected score maximization routine. ± indicates the standard deviation of the result under bootstrap resamplings.