Parameter Learning and Convergent Inference for Dense Random Fields

Philipp Krähenbühl PHILKR@CS.STANFORD.EDU
Vladlen Koltun VLADLEN@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305 USA

Abstract

Dense random fields are models in which all pairs of variables are directly connected by pairwise potentials. It has recently been shown that mean field inference in dense random fields can be performed efficiently and that these models enable significant accuracy gains in computer vision applications. However, parameter estimation for dense random fields is still poorly understood. In this paper, we present an efficient algorithm for learning parameters in dense random fields. All parameters are estimated jointly, thus capturing dependencies between them. We show that gradients of a variety of loss functions over the mean field marginals can be computed efficiently. The resulting algorithm learns parameters that directly optimize the performance of mean field inference in the model. As a supporting result, we present an efficient inference algorithm for dense random fields that is guaranteed to converge.

1. Introduction

Random field models are often used to express image coherence priors in computer vision applications, such as segmentation and reconstruction (Blake et al., 2011). Most random field models explored in the literature are sparsely connected, with the most common structure being a grid. Such models can lead to excessive smoothing of object boundaries and to inconsistencies over longer ranges in the image. Models with hierarchical structure and higher-order potentials have been proposed to reduce these artifacts (He et al., 2004; Roth & Black, 2009; Kohli et al., 2009). Nevertheless, the connectivity of the models remained relatively sparse due to the limitations of available inference algorithms.

Recently, Krähenbühl and Koltun (2011) introduced an efficient algorithm for performing mean field inference in fully connected random fields. Such dense random fields support explicit modeling of long-range correlations in the image and have enabled substantial accuracy gains in pixel-level labeling problems. The algorithm of Krähenbühl and Koltun handles pairwise potentials that can be expressed in terms of Gaussian kernels in feature space, and inference is performed by means of rapid approximate Gaussian convolutions. Vineet et al. (2012a; 2012b) have extended convolutional inference to more general Gaussian kernels and to models with higher-order terms.

Despite the development of efficient inference algorithms, parameter estimation in dense random fields is still poorly understood. Prior works use piecewise training, in which different types of parameters are learned separately. For example, parameters for the unary potentials and for the pairwise potentials are separately estimated (Krähenbühl & Koltun, 2011; Vineet et al., 2012a;b). Krähenbühl and Koltun estimate some parameters using grid search, and Vineet et al. use a simplified generative model with additional independence assumptions.

In this paper, we present a general algorithm for parameter estimation in dense random fields. All parameters are learned jointly, thus capturing dependencies between them. The algorithm supports a variety of loss functions formulated over the mean field marginals. The learning algorithm thus estimates parameters that optimize the performance of mean field inference.

Algorithms for parameter estimation in random field models minimize a loss function, such as the negative log-likelihood (LeCun et al., 2006; Koller & Friedman, 2009; Tappen, 2011). This is commonly done using gradient-based optimization. Computing the exact gradient is computationally intractable in general

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

because it requires performing exact inference in the model. One way to deal with this is to approximate the gradient by substituting approximate inference in place of exact inference (Wainwright et al., 2003; Sutton & McCallum, 2005; Kumar et al., 2005; Vishwanathan et al., 2006; Levin & Weiss, 2009; Pal et al., 2012). While it can be shown that the approximate gradient obtained in this way corresponds to a well-defined surrogate objective (Wainwright, 2006), it may not optimize the performance of the learned model on the data (Kulesza & Pereira, 2007).

We take a different approach and directly minimize a loss function on the approximate distribution that is used by the inference algorithm (Tappen, 2007; Samuel & Tappen, 2009). This is known as marginal-based loss (Domke, 2013). The advantage is that the performance of the inference algorithm is optimized directly: parameters are learned so as to maximize the accuracy of the mean field approximation.

To optimize the marginal-based loss function we need to compute its gradient. This is challenging because there is no closed-form expression for the mean field marginals. However, the marginals can be defined recursively, as the product of iterative message passing. Using this definition, we derive an efficient algorithm for computing the gradient of the marginal-based loss function over the parameters of the model.

As a supporting result, we develop an efficient mean field inference algorithm for dense random fields that is guaranteed to converge. Mean field inference is known to converge if message passing is performed sequentially (Wainwright & Jordan, 2008). However, for dense random fields, sequential message passing is impractical due to the huge number of pairwise connections. The convolutional inference of Krähenbühl and Koltun performs all message passing in the model in parallel. This is the key to its efficiency, but it also invalidates the traditional convergence guarantees. We present a different convolutional inference algorithm that is guaranteed to converge.

2. Preliminaries

Model. The dense conditional random field is defined over a set $\mathbf{X} = \{X_1, \dots, X_N\}$ of variables conditioned on the image $\mathbf{I}$ and the model parameters $\boldsymbol{\theta}$. The domain of each variable is a set $\mathcal{L} = \{l_1, l_2, \dots, l_M\}$ of labels. The Gibbs energy of a label assignment $\mathbf{x} \in \mathcal{L}^N$ is

$$E(\mathbf{x}|\mathbf{I}, \boldsymbol{\theta}) = \sum_i \psi_i(x_i|\boldsymbol{\theta}) + \sum_{i<j} \psi_{ij}(x_i, x_j|\boldsymbol{\theta}), \quad (1)$$

where i and j range from 1 to N. The unary potentials $\psi_i(x_i|\boldsymbol{\theta})$ and the pairwise potentials $\psi_{ij}(x_i, x_j|\boldsymbol{\theta})$ are implicitly conditioned on the image $\mathbf{I}$. Each variable X_i is associated with a fixed feature vector $\mathbf{f}_i$, determined by the image $\mathbf{I}$. The pairwise potentials are modeled as mixtures of kernels in feature space:

$$\psi_{ij}(x_i, x_j|\boldsymbol{\theta}) = \sum_{m=1}^{C} \mu^{(m)}(x_i, x_j|\boldsymbol{\theta}) k^{(m)}(\mathbf{f}_i - \mathbf{f}_j). \quad (2)$$

Here $\mu^{(m)}$ is a label compatibility function that measures how likely two classes are to occur near each other. The simplest such label compatibility is the Potts model: $\mu^{(m)}(x_i, x_j|\boldsymbol{\theta}) = 1_{[x_i \neq x_j]}$.

Inference. Mean field inference computes a distribution $Q(\mathbf{X})$ that best approximates the probability distribution $P(\mathbf{X}|\mathbf{I}, \boldsymbol{\theta}) = \frac{1}{Z(\mathbf{I}, \boldsymbol{\theta})} \exp(-E(\mathbf{X}|\mathbf{I}, \boldsymbol{\theta}))$ of the model. $Q(\mathbf{X}) = \prod_i Q_i(X_i)$ is a product of independent marginals over each of the variables. Each of the marginals is constrained to be a proper probability distribution: $\sum_{x_i} Q_i(x_i) = 1$, $Q_i(X_i) \geq 0$. The mean field approximation minimizes the KL divergence

$$\begin{aligned} &D(Q\|P) = \\ &\sum_{i,x_i} Q_i(x_i) \log Q_i(x_i) + \sum_{i,x_i} \psi_i(x_i|\boldsymbol{\theta}) Q_i(x_i) + \\ &\sum_{i<j} \sum_{x_i, x_j} \psi_{ij}(x_i, x_j|\boldsymbol{\theta}) Q_i(x_i) Q_j(x_j) + \log Z(\mathbf{I}, \boldsymbol{\theta}). \end{aligned} \quad (3)$$

Traditional mean field inference performs the following message passing update on each marginal Q_i in turn until convergence:

$$Q_i(x_i) = \frac{1}{Z_i} \exp\Big(-\psi_i(x_i|\boldsymbol{\theta}) - \sum_{j \neq i} \sum_{x_j} \psi_{ij}(x_i, x_j|\boldsymbol{\theta}) Q_j(x_j)\Big),$$

where Z_i is the marginal partition function. Each update is guaranteed to decrease the KL divergence and this inference algorithm is guaranteed to converge to a local optimum (Wainwright & Jordan, 2008; Koller & Friedman, 2009).

In dense random fields the computational bottleneck is the evaluation of the sum $\sum_{j \neq i} \sum_{x_j} \psi_{ij}(x_i, x_j|\boldsymbol{\theta}) Q_j(x_j)$. The computational complexity of a single update of a marginal $Q_i(X_i)$ is $O(N)$ and the complexity of updating all the marginals is $O(N^2)$.

Krähenbühl and Koltun (2011) observed that a high dimensional Gaussian filter can be used to update all the mean field marginals concurrently in time $O(N)$. This makes inference tractable. Unfortunately, the convergence guarantees traditionally associated with mean field inference break down when message passing is performed in parallel. In the next section, we

present two variants of a new efficient inference algorithm in dense random fields that is guaranteed to converge for a broad class of kernels and label compatibility functions.

3. Convergent convolutional inference

We begin with two sufficient conditions under which we can perform parallel message passing and still guarantee convergence. The first condition is that all kernels $k^{(m)}$ must be positive definite. A kernel k is defined to be positive definite if and only if for an arbitrary set of feature vectors $\{\mathbf{f}_1, \ldots, \mathbf{f}_N\}$ the kernel matrix $\mathbf{K}$ with entries $\mathbf{K}_{ij} = k(\mathbf{f}_i - \mathbf{f}_j)$ is positive definite. Bochner's theorem implies that any kernel $k^{(m)}$ with a non-negative Fourier spectrum is positive definite. Positive definite kernels are also known as Mercer kernels and $\mathbf{K}$ is their Gram matrix. The Gaussian kernel $k(\mathbf{x}) = e^{-\frac{1}{2}\mathbf{x}^2}$, the tent filter $k(\mathbf{x}) = \max(1 - \|\mathbf{x}\|_1, 0)$, and the sinc filter $k(x) = \frac{\sin(x)}{x}$ are examples of positive definite kernels. Note that conventional grid-structured random field models are a special case of the tent filter; in this case $k(\mathbf{f}_i - \mathbf{f}_j) = \max(1 - \frac{1}{2}\|\mathbf{f}_i - \mathbf{f}_j\|_1, 0)$, where $\mathbf{f}_i$ is the two-dimensional coordinate vector of pixel X_i.

The second convergence condition is that each label compatibility function $\mu^{(m)}$ is negative semidefinite. A label compatibility function μ is defined to be negative semidefinite if and only if there exists a constant c for which the matrix $\boldsymbol{\mu}$ with entries $\boldsymbol{\mu}_{ij} = \mu(l_i, l_j) + c$ is negative semidefinite. Note that adding the constant c to all label compatibility values simply reparameterizes the Gibbs energy of the random field and does not change the modeled probability distribution. The Potts model $\mu(l_i, l_j) = 1_{[l_i \neq l_j]}$ is negative semidefinite with $c = -1$. Other negative semidefinite label compatibility functions include negative diagonal models $\mu(l, l) = -\alpha_l$, negative diagonally dominant models $\mu(l_i, l_i) < -\sum_{l_j \neq l_i} |\mu(l_i, l_j)|$, and the L^1 norm $\mu(l_i, l_j) = |l_i - l_j|$.

We now present an efficient parallel mean field inference algorithm for dense random fields that is guaranteed to converge for models with positive definite kernels and negative semidefinite label compatibility functions.

3.1. Convergent parallel mean field

We begin by rewriting the KL divergence (3) in vector notation:

$$D(Q\|P) = \underbrace{\mathbf{q}^\top \log \mathbf{q}}_{\text{entropy}} + \underbrace{\mathbf{q}^\top \mathbf{u} + \frac{1}{2}\mathbf{q}^\top \Psi \mathbf{q} + \log Z(\mathbf{I}, \boldsymbol{\theta})}_{\text{cross entropy}}, \quad (4)$$

where $\mathbf{q}$ and $\mathbf{u}$ are vectors of length $N \times M$ with elements $\mathbf{q}_{(i,l)} = Q_i(X_i = l)$ and $\mathbf{u}_{(i,l)} = \psi_i(X_i = l)$, respectively. We use the tuple (i, l) to refer to variable X_i and label l. For notational convenience we use $\mathbf{q}_i$ to denote a vector of the M elements in $\mathbf{q}$ associated with variable X_i. The vector $\mathbf{u}_i$ is defined analogously. The pairwise potentials make up the symmetric matrix Ψ with entries $\Psi_{(i,l),(j,k)} = \psi_{ij}(X_i = l, X_j = k)$. Since there is no pairwise term between a random variable X_i and itself, Ψ has a zero block diagonal. Equivalently, we can express Ψ as the Kronecker product of the kernel matrix $\mathbf{K}$ and the label compatibility matrix $\boldsymbol{\mu}$:

$$\Psi = \sum_{m=1}^{C} \left(\mathbf{K}^{(m)} - \mathbf{I}_N\right) \otimes \boldsymbol{\mu}^{(m)}. \quad (5)$$

For notational simplicity we assume that the kernel matrix $\mathbf{K}^{(m)}$ has a unit diagonal: $\mathbf{K}^{(m)}_{ii} = k^{(m)}(\mathbf{0}) = 1$. Subtracting the identity matrix $\mathbf{I}_N$ from $\mathbf{K}^{(m)}$ in the above equation yields a zero block diagonal in Ψ.

For a negative semidefinite label compatibility function $\mu^{(m)}$ and a positive definite kernel $k^{(m)}$ the Kronecker product $\mathbf{K}^{(m)} \otimes \boldsymbol{\mu}^{(m)}$ is negative semidefinite and hence concave. The block diagonal matrix $-\mathbf{I}_N \otimes \boldsymbol{\mu}^{(m)}$ is positive definite and convex. This allows us to write the KL divergence as the sum of a convex function $f(\mathbf{q})$ and a concave function $g(\mathbf{q})$:

$$f(\mathbf{q}) = \mathbf{q}^\top \log \mathbf{q} - \frac{1}{2} q^\top \left(\sum_{m=1}^{C} \mathbf{I}_N \otimes \boldsymbol{\mu}^{(m)}\right) \mathbf{q}$$

$$g(\mathbf{q}) = \mathbf{q}^\top \mathbf{u} + \frac{1}{2}\mathbf{q}^\top \left(\sum_{m=1}^{C} \mathbf{K}^{(m)} \otimes \boldsymbol{\mu}^{(m)}\right) \mathbf{q} + \log Z(\mathbf{I}, \boldsymbol{\theta}).$$

Note that the concave function $g(\mathbf{q})$ includes all the pairwise interactions between different marginals $\mathbf{q}_i$ and $\mathbf{q}_j$, while $f(\mathbf{q})$ only involves local terms over individual marginals $\mathbf{q}_i$.

We use the concave-convex procedure (CCCP) to minimize the KL divergence (Yuille & Rangarajan, 2001). CCCP iteratively minimizes a series of energy functions $f(\mathbf{q}^{(t)}) + \mathbf{q}^{(t)\top} \nabla g\left(\mathbf{q}^{(t-1)}\right)$ until convergence, where $\mathbf{q}^{(t)}$ and $\mathbf{q}^{(t-1)}$ are the mean field marginals at iteration t and $t-1$, respectively. CCCP is guaranteed to converge to a local minimum (Sriperumbudur & Lanckriet, 2009).

CCCP leads to the following optimization problem:

$$\begin{aligned} \underset{\mathbf{q}}{\text{minimize}} \quad & \mathbf{q}^\top \log \mathbf{q} + \mathbf{q}^\top \mathbf{e}^{(t)} - \frac{1}{2} \sum_{m=1}^{C} \mathbf{q}^\top \mathbf{I}_N \otimes \boldsymbol{\mu}^{(m)} \mathbf{q} \\ \text{subject to} \quad & \mathbf{1}^\top \mathbf{q}_i = 1 \quad \text{for } i = 1, \ldots, N \end{aligned} \quad (6)$$

where $\mathbf{e}^{(t)} = \nabla g(\mathbf{q}^{(t-1)})$. The entropy term $\mathbf{q}^\top \log \mathbf{q}$ restricts the domain of $\mathbf{q}$ to non-negative values. This makes an explicit constraint $\mathbf{q} \geq 0$ redundant.

For each variable X_i, the gradient is defined as

$$\mathbf{e}_i^{(t)} = \mathbf{u}_i + \sum_{m=1}^{C} \sum_j k^{(m)}(\mathbf{f}_i - \mathbf{f}_j)\boldsymbol{\mu}^{(m)}\mathbf{q}_j^{(t-1)}. \quad (7)$$

This is almost identical to message passing in Krähenbühl and Koltun's algorithm (2011). It is the combination of a matrix multiplication $\boldsymbol{\mu}^{(m)}\mathbf{q}_j^{(t-1)}$ and a convolution with a kernel $k^{(m)}$, both of which can be evaluated in linear time.

In the mean field optimization problem (6), each marginal $\mathbf{q}_i$ is independent of every other marginal $\mathbf{q}_j$. We minimize each marginal $\mathbf{q}_i$ independently. The Karush-Kuhn-Tucker (KKT) conditions are

$$\log \mathbf{q}_i^{(t)} - \left(\sum_{m=1}^{C} \boldsymbol{\mu}^{(m)}\right) \mathbf{q}_i^{(t)} + \lambda \mathbf{1} + \mathbf{e}_i^{(t)} = \mathbf{0}$$
$$\mathbf{1}^\top \mathbf{q}_i^{(t)} = 1. \quad (8)$$

Strong duality holds since the objective in (6) is convex and continuously differentiable, and the constraints are affine. This means that a solution to the KKT conditions (8) minimizes the objective (6). We solve the KKT conditions using Newton's method.

This algorithm is somewhat slower than the algorithm of Krähenbühl and Koltun since we need to solve a nonlinear system of equations rather than simply exponentiating and normalizing the marginals after each parallel message passing step. In Section 3.2 we present a slight modification of the mean field objective that leads to a faster and simpler convergent inference algorithm that produces almost identical results in practice.

3.2. Concave cross-entropy approximation

The cross-entropy term of the KL divergence in Equation 4 is close to concave, especially for large kernels $k^{(m)}$. The only source of non-concavity is the block diagonal matrix $\mathbf{I}_N \otimes \boldsymbol{\mu}$, which contributes very little to the pairwise term Ψ. For example, a two-dimensional Gaussian kernel of standard deviation $\geq$ 3px contains less than 1% of its mass at the center. In practice, the error introduced by simply ignoring the presence of the extraneous block diagonal is negligible, as we will verify in Section 5. This motivates the concave approximation of the cross entropy:

$$D(Q\|P) = \underbrace{\mathbf{q}^\top \log \mathbf{q}}_{\text{entropy}} + \underbrace{\mathbf{q}^\top \mathbf{u} + \frac{1}{2}\mathbf{q}^\top \hat{\Psi}\mathbf{q} + \log Z(\mathbf{I}, \boldsymbol{\theta})}_{\text{approximate cross entropy}}, \quad (9)$$

where $\hat{\Psi} = \sum_{m=1}^{C} \mathbf{K}^{(m)} \otimes \boldsymbol{\mu}^{(m)}$.

Equation 9 is the sum of a convex function, the entropy, and a concave function, the approximate cross entropy. We again use the concave-convex procedure. The gradient of the approximate cross entropy yields the same message passing step as Equation 7 and is again evaluated efficiently using high-dimensional filtering.

The CCCP algorithm then minimizes the convex entropy $\mathbf{q}^\top \log \mathbf{q}$ plus a linear term $\mathbf{q}^\top \mathbf{e}^{(t)}$ under the constraint that $\mathbf{q}$ is a proper probability distribution. We minimize the convex objective by solving the KKT conditions for each mean field marginal:

$$\log \mathbf{q}_i^{(t)} + \lambda \mathbf{1} + \mathbf{e}_i^{(t)} = \mathbf{0}$$
$$\mathbf{1}^\top \mathbf{q}_i^{(t)} = 1. \quad (10)$$

The solution is the standard mean field update $\mathbf{q}_i^{(t)} = \frac{1}{Z_i} \exp\left(-\mathbf{e}_i^{(t)}\right)$.

The resulting algorithm is almost identical to the algorithm of Krähenbühl and Koltun. Like the algorithm in Section 3.1, it is guaranteed to converge. However, since the KKT conditions have a closed-form solution, this algorithm is more than three times faster in practice than the algorithm in Section 3.1 and has analogous performance to the algorithm of Krähenbühl and Koltun, as shown in Section 5.

4. Parameter learning

We now derive the main result of this paper: a parameter learning algorithm for dense random fields. The algorithm minimizes a differentiable loss function $L(\mathbf{q})$ over the mean field marginals $\mathbf{q}$. A number of loss functions are described in Section 4.3.

It is easy to see that the marginals $\mathbf{q}$ depend on the model parameters $\boldsymbol{\theta}$. For example, changing the parameters of the unary potentials ψ_i can move the mode of the variational approximation and change the marginal distributions $\mathbf{q}$ obtained by the inference procedure. Let's make this functional dependence explicit: $\mathbf{q} = \mathbf{q}(\boldsymbol{\theta})$. As shown in supplementary material, $\mathbf{q}(\boldsymbol{\theta})$ is continuously differentiable. Differentiability implies that for any pair of sufficiently similar model parameter vectors, the mean field approximations are also similar: no pair of sufficiently close parameter vectors can lead the mean field inference into different local optima.

We use gradient-based optimization to minimize $L(\mathbf{q}(\boldsymbol{\theta}))$. For the remainder of this section we assume that inference is performed using the algorithm

described in Section 3.2. The algorithm described in Section 3.1 and the original algorithm of Krähenbühl and Koltun can be handled analogously.

4.1. Mean field gradient

The gradient of the marginal-based loss is

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}^{\top}. \tag{11}$$

We use implicit differentiation of the KKT conditions (10) to derive the gradient $\frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}$ of the mean field approximation. The gradient $\frac{\partial L}{\partial \mathbf{q}}$ of the loss function will be treated in Section 4.3.

The derivative of the KKT conditions with respect to the model parameters yields a set of linear constraints on the gradient of each marginal:

$$\begin{aligned} \operatorname{diag}\left(\frac{1}{\mathbf{q}_i^{(t)}}\right) \frac{\partial \mathbf{q}_i^{(t)}}{\partial \boldsymbol{\theta}} &= -\frac{\partial \lambda}{\partial \boldsymbol{\theta}} \mathbf{1} - \frac{\partial \mathbf{e}_i^{(t)}}{\partial \boldsymbol{\theta}} \\ \mathbf{1}^{\top} \frac{\partial \mathbf{q}_i^{(t)}}{\partial \boldsymbol{\theta}} &= 0. \end{aligned} \tag{12}$$

As shown in supplementary material, the linear system (12) is full rank and invertible. The gradient $\frac{\partial \mathbf{q}_i^{(t)}}{\partial \boldsymbol{\theta}}$ is thus always well-defined and is given by

$$\frac{\partial \mathbf{q}_i^{(t)}}{\partial \boldsymbol{\theta}} = A_i^{(t)} \frac{\partial \mathbf{e}_i^{(t)}}{\partial \boldsymbol{\theta}}, \tag{13}$$

where $A_i^{(t)} = \mathbf{q}_i^{(t)} \mathbf{q}_i^{(t)\top} - \operatorname{diag}\left(\mathbf{q}_i^{(t)}\right)$. For notational convenience let $A^{(t)}$ be the block diagonal matrix with blocks $A_i^{(t)}$, for $i = 1, \ldots, N$.

We compute the gradient of the message passing term $\mathbf{e}^{(t)}$ using the product rule:

$$\frac{\partial \mathbf{e}^{(t)}}{\partial \boldsymbol{\theta}} = \hat{\Psi} \frac{\partial \mathbf{q}^{(t-1)}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{u}}{\partial \boldsymbol{\theta}} + \mathbf{q}^{(t-1)} \frac{\partial \hat{\Psi}}{\partial \boldsymbol{\theta}}. \tag{14}$$

The gradient $\frac{\partial \mathbf{e}^{(t)}}{\partial \boldsymbol{\theta}}$ is recursively defined in terms of the gradient $\frac{\partial \mathbf{q}^{(t-1)}}{\partial \boldsymbol{\theta}}$ of the marginals from the previous iteration, the unary gradient $\frac{\partial \mathbf{u}}{\partial \boldsymbol{\theta}}$, and the pairwise gradient tensor $\frac{\partial \hat{\Psi}}{\partial \boldsymbol{\theta}}$. Unary and pairwise gradients with respect to various model parameters are derived in Section 4.2.

Evaluating equations 13 and 14 directly involves n message passing steps per parameter. Computationally this is as expensive as computing the numeric gradient using finite differencing. For models with more than a few parameters this is infeasible.

While a direct evaluation of the gradient $\frac{\partial \mathbf{q}_i^{(t)}}{\partial \boldsymbol{\theta}}$ is impractical, we can still evaluate the gradient $\frac{\partial L}{\partial \boldsymbol{\theta}}$ efficiently. We recursively substitute the gradient of the mean field marginals (13) and the gradient of the message passing term (14) back into Equation 11. The resulting gradient expression is

$$\frac{\partial L(\mathbf{q}^{(n)}(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{n} \mathbf{b}^{(t)\top} \left(\frac{\partial \mathbf{u}}{\partial \boldsymbol{\theta}} + \mathbf{q}^{(t-1)} \frac{\partial \hat{\Psi}}{\partial \boldsymbol{\theta}} \right), \tag{15}$$

where $\mathbf{b}^{(t)} = A^{(t)} \hat{\Psi} \mathbf{b}^{(t+1)}$ is the normalized loss gradient at iteration t and $\mathbf{b}^{(n)} = A^{(n)} \left(\nabla L(\mathbf{q}^{(n)})\right)^{\top}$. Unlike a direct evaluation, this back substitution leads to an algorithm that requires only n parallel message passing steps in total, independent of the number of parameters. The gradient computation is summarized in Algorithm 1.

Algorithm 1 Mean field gradient

1: **for** $t = 1$ **to** n **do**
2: Run inference to compute $\mathbf{q}^{(t)}$
3: **end for**
4: $\mathbf{b}^{(n)} \leftarrow A^{(n)} \left(\nabla L(\mathbf{q}^{(n)})\right)^{\top}$
5: **for** $t = n - 1$ **to** 1 **do**
6: $\mathbf{b}^{(t)} \leftarrow A^{(t)} \hat{\Psi} \mathbf{b}^{(t+1)}$ ▷ Message passing
7: **end for**
8: $\mathbf{g} \leftarrow \mathbf{0}$
9: **for** $t = 1$ **to** n **do**
10: $\mathbf{g} \leftarrow \mathbf{g} + \frac{\partial \mathbf{b}^{(t)\top} \mathbf{u}}{\partial \boldsymbol{\theta}}$ ▷ Unary gradient
11: $\mathbf{g} \leftarrow \mathbf{g} + \frac{\partial \mathbf{b}^{(t)\top} \hat{\Psi} \mathbf{q}^{(t)}}{\partial \boldsymbol{\theta}}$ ▷ Pairwise gradient
12: **end for**
13: $\frac{\partial L(\mathbf{q}^{(n)})}{\partial \boldsymbol{\theta}} \leftarrow \mathbf{g}$

This algorithm is closely related to back belief propagation (Eaton & Ghahramani, 2009) and back tree-reweighted belief propagation (Domke, 2013).

Algorithm 1 begins by performing inference to compute the mean field marginals $\mathbf{q}^{(t)}$ for every iteration t (lines 1 - 3). Next, message passing is used to compute the normalized loss gradient $\mathbf{b}^{(t)}$ (lines 4 - 7). This message passing step is similar to inference, but instead of exponentiating and normalizing we simply multiply by the matrix $A^{(t)}$. Finally, the gradient of the objective is computed as the sum of unary and pairwise gradients (lines 10 and 11). We will now show how to evaluate these gradients efficiently.

4.2. Unary and pairwise gradients

In this section, we describe how to efficiently evaluate the gradient terms in lines 10 and 11 of Algorithm 1 with respect to various model parameters. Let $\mathbf{b}^{(t)}$ and $\mathbf{q}^{(t)}$ be given. For notational simplicity we drop the superscript for the remainder of this section.

Logistic parameters. We assume that the unary term is a logistic regression model with $\mathbf{u}_i = \mathbf{\Lambda}\mathbf{h}_i$ for a parameter matrix $\mathbf{\Lambda}$ and features $\mathbf{h}_i$ for variable X_i. (In our experiments, $\mathbf{h}_i$ is the result of a TextonBoost classifier at pixel i (Shotton et al., 2009).) The unary gradient with respect to the logistic parameters $\mathbf{\Lambda}$ is given by

$$\frac{\partial \mathbf{b}^\top \mathbf{u}}{\partial \mathbf{\Lambda}} = \sum_i \mathbf{b}_i \mathbf{h}_i^\top.$$

Label compatibility. Recall the definition of the pairwise term $\hat{\Psi} = \sum_{m=1}^{C} \mathbf{K}^{(m)} \otimes \boldsymbol{\mu}^{(m)}$. The pairwise gradient with respect to the label compatibility parameters is given by

$$\frac{\partial \mathbf{b}^\top \hat{\Psi}\mathbf{q}}{\partial \boldsymbol{\mu}^{(m)}} = \sum_i \mathbf{b}_i \left(\sum_j \mathbf{K}_{ij}^{(m)} \mathbf{q}_j^\top \right).$$

The second summation is simply a convolution of the marginals $\mathbf{q}$ with kernel $k^{(m)}$. As with message passing, this convolution can be performed efficiently and the gradient can be computed in linear time. Note that we usually require μ to be symmetric; this is enforced by setting $\mu(a, b) = \mu(b, a)$ to be a single variable and adapting the gradient expression accordingly.

Kernel shape. Consider a Gaussian kernel of the form $k(\mathbf{f}_i - \mathbf{f}_j) = e^{-\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)\Sigma^{-1}(\mathbf{f}_i - \mathbf{f}_j)}$. The gradient of this function with respect to its parameters is

$$\frac{\partial k(\mathbf{f}_i - \mathbf{f}_j)}{\partial \Sigma} = \frac{1}{2}\Sigma^{-1}(\mathbf{f}_i - \mathbf{f}_j)(\mathbf{f}_i - \mathbf{f}_j)^\top \Sigma^{-1} k(\mathbf{f}_i - \mathbf{f}_j).$$

This expression is no longer Gaussian. However, we can express it as a weighted sum of Gaussians and evaluate it efficiently. Define $\tilde{\mathbf{f}}_i = \Sigma^{-1}\mathbf{f}_i$. The pairwise gradient with respect to the kernel parameters Σ is

$$\frac{\partial \mathbf{b}^\top \hat{\Psi}\mathbf{q}}{\partial \mathbf{\Sigma}^{(m)}} =$$
$$\frac{1}{2}\sum_i \tilde{\mathbf{f}}_i \left(\mathbf{b}_i^\top \sum_j \mathbf{K}_{ij}^{(m)} \mathbf{q}_j + \mathbf{q}_i^\top \sum_j \mathbf{K}_{ji}^{(m)} \mathbf{b}_j \right) \tilde{\mathbf{f}}_i^\top -$$
$$\frac{1}{2}\sum_i \tilde{\mathbf{f}}_i \left(\mathbf{b}_i^\top \sum_j \mathbf{K}_{ij}^{(m)} \mathbf{q}_j \tilde{\mathbf{f}}_j^\top + \mathbf{q}_i^\top \sum_j \mathbf{K}_{ji}^{(m)} \mathbf{b}_j \tilde{\mathbf{f}}_j^\top \right).$$

Each of the four summations over j is a high-dimensional Gaussian convolution and can be performed efficiently. In order to compute this gradient we perform a total of $2M(D + 1)$ convolutions, where M is the number of labels and D is the dimensionality of the feature vector. The total running time is linear in the number of variables N.

4.3. Loss functions

We now consider a number of loss functions $L(\mathbf{q})$ defined over the mean field marginals $\mathbf{q}$ and derive their analytic gradient $\nabla L(\mathbf{q})$. Each of the loss functions is formulated in terms of a ground truth labeling $\mathcal{T} \in \mathcal{L}^N$ for an individual training image.

Log-likelihood. The commonly used log-likelihood loss is defined as

$$L_\ell(\mathbf{q}) = -\sum_i w_{\mathcal{T}_i} \log(\mathbf{q}_{(i,\mathcal{T}_i)}),$$

where $w_{\mathcal{T}_i}$ is a label-dependent weight that can normalize out imbalances in the available training data. The gradient of the log-likelihood loss is given by

$$\frac{\partial L_\ell}{\partial \mathbf{q}_{(i,l)}} = -\frac{w_{\mathcal{T}_i} 1_{[\mathcal{T}_i = l]}}{\mathbf{q}_{(i,l)}}.$$

The log-likelihood objective does not cope well with outliers or mislabeled variables. For example, improving the likelihood of a hopeless outlier from 10^{-5} to 10^{-4} reduces the loss more than improving the likelihood of a different data point from $\frac{1}{10}$ to $\frac{1}{2}$. To ameliorate this sensitivity to outliers, we can use a robust version of the log-likelihood.

Robust log-likelihood. The robust log-likelihood loss is defined as

$$L_r(\mathbf{q}) = -\sum_i w_{\mathcal{T}_i} \log(\mathbf{q}_{(i,\mathcal{T}_i)} + \epsilon).$$

This objective can be interpreted as a softmax of the log-likelihood loss with a threshold $-\log(\epsilon)$. In this formulation, the penalty incurred by outliers with extremely low likelihoods is fixed. The gradient of the robust log likelihood objective is

$$\frac{\partial L_r}{\partial \mathbf{q}_{(i,l)}} = -\frac{w_{\mathcal{T}_i} 1_{[\mathcal{T}_i = x_i]}}{\mathbf{q}_{(i,l)} + \epsilon}.$$

Hamming loss. The log-likelihood and robust log-likelihood objectives may not reflect performance measures that can be relevant for particular computer vision tasks. For example, for a per-pixel labeling task, we may want to maximize global labeling accuracy, class-average accuracy, or another measure of labeling performance such as intersection over union. Our approach can incorporate such performance measures directly, by explicitly casting them as marginal-based loss objectives that should be optimized by the learning algorithm.

We first consider global and class-average accuracy. These can both be cast as weighted versions of the

Hamming loss $h(\mathbf{x}) = \sum_i w_{\mathcal{T}_i} 1_{[\mathbf{x}_i \neq \mathcal{T}_i]}$, which counts the number of incorrect labels in a label assignment $\mathbf{x}$. For $w_l = 1$, the Hamming loss corresponds to global labeling accuracy. Setting $w_l = \frac{1}{\sum_i [l=\mathcal{T}_i]}$ leads to the class-average accuracy measure. We can also use a weighted geometric mean to interpolate between the two objectives.

The Hamming loss is not continuous and is not immediately amenable to gradient-based optimization. We use the expected Hamming loss instead:

$$L_h(\mathbf{q}) = C - \sum_i w_{\mathcal{T}_i} \mathbf{q}_{(i,\mathcal{T}_i)},$$

where $C = \sum_i w_{\mathcal{T}_i}$ is a constant. The corresponding gradient is given by

$$\frac{\partial L_h}{\partial \mathbf{q}_{(i,l)}} = -w_{\mathcal{T}_i} 1_{[\mathcal{T}_i = l]}.$$

Intersection over union. The intersection-over-union objective can be used to correct for imbalances in the training data. The discrete intersection-over-union score is defined as

$$iu(\mathbf{x}) = \sum_l \frac{i_l(\mathbf{x})}{u_l(\mathbf{x})} = \sum_l \frac{\sum_i 1_{[\mathcal{T}_i = \mathbf{x}_i = l]}}{n_l + \sum_i 1_{[\mathcal{T}_i \neq l, \mathbf{x}_i = l]}},$$

where $n_l = \sum_i 1_{[\mathcal{T}_i = l]}$ is the number of variables with label l in the training data. The intersection-over-union score is evaluated over the complete dataset, which sets it apart from the preceding objectives. We formulate a relaxed marginal-based intersection-over-union loss:

$$L_{iu}(\mathbf{q}) = -\sum_l \frac{i_l(\mathbf{q})}{u_l(\mathbf{q})} = -\sum_l \frac{\sum_i 1_{[\mathcal{T}_i = l]} \mathbf{q}_{(i,l)}}{n_l + \sum_i 1_{[\mathcal{T}_i \neq l]} \mathbf{q}_{(i,l)}}.$$

$L_{iu}(\mathbf{q})$ ties together all the marginals for all the images in the training set, not just a single image. Despite this global interdependence, the gradient is simple:

$$\frac{\partial L_{iu}}{\partial \mathbf{q}_{(i,l)}} = \begin{cases} -\frac{1}{u_l(\mathbf{q})} & \text{if } \mathcal{T}_i = l \\ \frac{i_l(\mathbf{q})}{u_l(\mathbf{q})^2} & \text{otherwise} \end{cases}.$$

In fact, once the intersection and union values i_l and u_l are known the partial derivative of the intersection-over-union loss is independent for every variable X_i.

5. Experiments

We build on the publicly available implementation of Krähenbühl and Koltun (2011). The original implementation uses normalized filter kernels $\tilde{k}(\mathbf{f}_i, \mathbf{f}_j) = \frac{k(\mathbf{f}_i, \mathbf{f}_j)}{\sum_j k(\mathbf{f}_i, \mathbf{f}_j)}$. This normalization greatly increases the accuracy of the approximate filtering, but it breaks the symmetry assumptions of the model since $\tilde{k}(\mathbf{f}_i, \mathbf{f}_j)$ and $\tilde{k}(\mathbf{f}_j, \mathbf{f}_i)$ are normalized differently. We use a symmetric normalization instead: $\tilde{k}(\mathbf{f}_i, \mathbf{f}_j) = \frac{k(\mathbf{f}_i, \mathbf{f}_j)}{\sqrt{\sum_j k(\mathbf{f}_i, \mathbf{f}_j) \sum_i k(\mathbf{f}_i, \mathbf{f}_j)}}$. This ensures that the kernel is symmetric and positive definite while keeping the approximation error of the filtering data structure low.

We evaluated the learning algorithm on the Pascal VOC 2010 dataset with the publicly available unary potentials and the training/validation/test split of Krähenbühl and Koltun. All parameters are learned on the validation set. We use the non-linear conjugate gradient algorithm for gradient-based optimization. We found that it performs better than L-BFGS for non-convex learning objectives.

We use a class weight $w_l = n_l^{-0.25}$ for all loss functions, where n_l is the number of pixels with label l in the training set. To prevent overfitting, we use a weak L2 regularizer. All experiments are performed on a single core of a 3.2GHz Intel Core i7 CPU.

5.1. Inference

We begin by comparing the performance of the convergent inference algorithm presented in Section 3.1, the faster convergent algorithm of Section 3.2, and the original inference algorithm of Krähenbühl and Koltun. For this experiment we use a Potts model with a single Gaussian kernel with spatial standard deviation 40px, color standard deviation 15, and interaction penalty 5. This is consistent with the settings used by Krähenbühl and Koltun on the Pascal VOC dataset. All inference algorithms were run for five iterations.

The three algorithms produce almost identical results. The difference in the inferred marginal probabilities, averaged over all images, pixels and labels is less than 2×10^{-5} between each pair of algorithms. On average, for a given pair of algorithms, less than 0.013% of the pixels in an image have a different MAP assignment. On the other hand, the running times differ significantly. The convergent parallel mean field algorithm of Section 3.1 takes 2.49s per image, while the convergent approximate algorithm of Section 3.2 takes 0.69s on average, which is slightly faster than the algorithm of Krähenbühl and Koltun (0.72s).

5.2. Learning

We now evaluate the parameter learning algorithm.

Label compatibility learning. In the first experiment, we focus on learning the label compatibility parameters. An approximate MLE algorithm for learn-

Unary potentials (no learning)	27.65%
Approximate gradient	30.16%
Our approach, log-likelihood loss	29.51%
Our approach, robust log-likelihood, $\epsilon = 0.1$	30.35%
Our approach, Hamming loss	30.70%
Our approach, intersection-over-union loss	**30.80%**

Table 1. Label compatibility learning. Comparison of the learning algorithm of Krähenbühl and Koltun, which follows an approximate log-likelihood gradient, and our approach, which evaluates an analytic gradient and admits a variety of learning objectives.

ing these parameters was described by Krähenbühl and Koltun. We compare their approach to our algorithm using a single-kernel model. For this experiment, we trained the kernel parameters using grid search and kept them fixed, to match the setting of the prior work. Our algorithm converged in 20-30 iterations. The runtime of each iteration is equivalent to two inference passes over the dataset.

The results are shown in Table 1. We observe significant differences between the various objectives supported by our algorithm. The straightforward log-likelihood loss is very sensitive to outliers: while the log-likelihood objective improved by several orders of magnitude during the optimization, this improvement is not reflected in the accuracy of the model. The robust log-likelihood reduces this sensitivity and significantly improves the results. It also outperforms the approximate MLE approach. Nevertheless, the log-likelihood in general does not correspond to the evaluation metric by which performance on the dataset is measured. The Hamming loss and the intersection-over-union loss better model the evaluation metric and offer additional performance improvements.

Kernel parameter learning. In the next experiment, we learn the kernel parameters, specifically the spatial and color standard deviations of the kernel. Krähenbühl and Koltun (2011) used grid search to estimate the kernel parameters and Vineet et al. (2012a) described a generative approach for this estimation task. To evaluate the different approaches, we train a Potts model with a single interaction penalty and five parameters for the Gaussian kernel: the standard deviations in the image coordinates and the color channels.

In evaluating grid search, we estimate only three parameters due to the exponential computational complexity of this approach: the Potts penalty, an isotropic spatial standard deviation, and an isotropic color deviation. (This mirrors the setup of Krähenbühl and Koltun.) Both grid search and our algorithm are able to learn the kernel parameters jointly with the weight of the Potts model. They also take the unary terms into account during learning and find jointly optimal parameters. The generative approach learns the five kernel parameters independently; we then estimate an optimal weight for the Potts model using grid search.

Unary potentials	No learning	27.65%
Potts model	Generative learning	28.01%
	Grid search	29.02%
	Our algorithm	**29.26%**
With label compatibility	Our algorithm	**31.13%**

Table 2. Kernel parameter learning. Comparative evaluation of the generative learning of Vineet et al., the brute force search of Krähenbühl and Koltun, and our algorithm.

The results are shown in Table 2. Our algorithm and grid search significantly outperform the generative learning framework. As further shown in the table, our algorithm is also able to learn the label compatibility parameters jointly with the kernel parameters, which increases the performance of the model by 2%.

Full model. In the final experiment we learn the parameters of the unary potentials jointly with the parameters of the pairwise potentials. These include the logistic regression parameters in the unary potentials, the label compatibility parameters, and the kernel parameters. The total number of learned parameters in the model is close to 700. To prevent overfitting, we use strong ridge regularization on the logistic regression parameters and on the off-diagonal entries of the label compatibility matrix. The accuracy of the learned model is **31.65%**, which is a 1.5% increase over the model of Krähenbühl and Koltun, a 4% increase over the unary potentials with a simple logistic regression model, and almost 10% over the raw TextonBoost features. Note that we did not include any additional information, such as dense SiftFlow initialization or bounding box detection (Vineet et al., 2012a;b).

6. Conclusion

We presented efficient convergent inference and learning algorithms for dense random fields. The learning algorithm jointly optimizes all parameters in the model and significantly outperforms alternative algorithms. Implementation of the presented algorithms will be made freely available.

Acknowledgements

Philipp Krähenbühl was supported by the Max Planck Center for Visual Computing and Communication.

References

Blake, Andrew, Kohli, Pushmeet, and Rother, Carsten. *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

Domke, Justin. Learning graphical model parameters with approximate marginal inference. *PAMI*, 2013. To appear.

Eaton, Frederik and Ghahramani, Zoubin. Choosing a variable to clamp. *JMLR*, 5:145–152, 2009.

He, Xuming, Zemel, Richard S., and Carreira-Perpinan, Miguel A. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

Kohli, Pushmeet, Ladický, Lubor, and Torr, Philip H. S. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3), 2009.

Koller, Daphne and Friedman, Nir. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Krähenbühl, Philipp and Koltun, Vladlen. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *NIPS*, 2011.

Kulesza, Alex and Pereira, Fernando. Structured learning with approximate inference. In *NIPS*, 2007.

Kumar, Sanjiv, August, Jonas, and Hebert, Martial. Exploiting inference for approximate parameter learning in discriminative fields. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*. 2005.

LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, Marc'Aurelio, and Huang, Fu Jie. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

Levin, Anat and Weiss, Yair. Learning to combine bottom-up and top-down segmentation. *IJCV*, 81: 105–118, 2009.

Pal, Christopher J., Weinman, Jerod J., Tran, Lam C., and Scharstein, Daniel. On learning conditional random fields for stereo. *IJCV*, 99(3):319–337, 2012.

Roth, Stefan and Black, Michael J. Fields of experts. *IJCV*, 82(2):205–229, 2009.

Samuel, Kegan G. G. and Tappen, Marshall F. Learning optimized MAP estimates in continuously-valued MRF models. In *CVPR*, 2009.

Shotton, Jamie, Winn, John M., Rother, Carsten, and Criminisi, Antonio. Textonboost for image understanding. *IJCV*, 81(1), 2009.

Sriperumbudur, Bharath K. and Lanckriet, Gert R. G. On the convergence of the concave-convex procedure. In *NIPS*, 2009.

Sutton, Charles A. and McCallum, Andrew. Piecewise training for undirected models. In *UAI*, 2005.

Tappen, Marshall F. Utilizing variational optimization to learn Markov random fields. In *CVPR*, 2007.

Tappen, Marshall F. Learning parameters in continuous-valued Markov random fields. In *Markov Random Fields For Vision And Image Processing*. MIT Press, 2011.

Vineet, Vibhav, Warrell, Jonathan, Sturgess, Paul, and Torr, Philip H. S. Improved initialization and Gaussian mixture pairwise terms for dense random fields with mean-field inference. In *BMVC*, 2012a.

Vineet, Vibhav, Warrell, Jonathan, and Torr, Philip H. S. Filter-based mean-field inference for random fields with higher-order terms and product label-spaces. In *ECCV*, 2012b.

Vishwanathan, S. V. N., Schraudolph, Nicol N., Schmidt, Mark W., and Murphy, Kevin P. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*, 2006.

Wainwright, Martin J. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *JMLR*, 7:1829–1859, 2006.

Wainwright, Martin J. and Jordan, Michael I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2), 2008.

Wainwright, Martin J., Jaakkola, Tommi S., and Willsky, Alan S. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *Proc. Workshop on Artificial Intelligence and Statistics*, 2003.

Yuille, Alan L. and Rangarajan, Anand. The concave-convex procedure (CCCP). In *NIPS*, 2001.