Journal of Machine Learning Research 17 (2016) 1-25 Submitted 6/13; Revised 6/16; Published 12/16

Blending Learning and Inference in Conditional Random Fields

Tamir Hazan TAMIR.HAZAN@TECHNION.AC.IL
Technion - Israel Institute of Technology
Technion City,
Haifa, 3200, Israel

Alexander G. Schwing ASCHWING@ILLINOIS.EDU
Electrical and Computer Engineering and Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Raquel Urtasun URTASUN@CS.TORONTO.EDU
University of Toronto
40 St. George Street,
Toronto, ON, M5S 2E4

Editor: Sebastian Nowozin

Abstract

Conditional random fields maximize the log-likelihood of training labels given the training data, e.g., objects given images. In many cases the training labels are structures that consist of a set of variables and the computational complexity for estimating their likelihood is exponential in the number of the variables. Learning algorithms relax this computational burden using approximate inference that is nested as a sub-procedure. In this paper we describe the objective function for nested learning and inference in conditional random fields. The devised objective maximizes the log-beliefs — probability distributions over subsets of training variables that agree on their marginal probabilities. This objective is concave and consists of two types of variables that are related to the learning and inference tasks respectively. Importantly, we afterwards show how to blend the learning and inference procedure and effectively get to the identical optimum much faster. The proposed algorithm currently achieves the state-of-the-art in various computer vision applications.

1. Introduction

Learning and inference of structured models drives much of the research in machine learning applications, from computer vision and natural language processing to computational biology. Examples include object detection (e.g., by Felzenszwalb et al. (2010)), parsing (e.g., by Koo et al. (2010)), or protein design (e.g., by Sontag et al. (2008)). The inference problem in these cases involves assessing the likelihood of labelings, whether outlined objects, parse trees, or molecular structures. The learning procedure searches for the parameters that maximize the likelihood of the training set.

Conditional random fields (CRFs) form an effective framework for maximizing the log-likelihood of training labels in structured models. Learning the parameters of these models

can be computationally expensive since the label space of real-world problems is usually exponential in the size of its variables.

When the label structure corresponds to a tree, exactly inferring the likelihood of the training labels can be done in time, linear in the number of variables using sum-product belief propagation as a subroutine. We refer to this approach as nested learning and inference. In contrast, when the label structure corresponds to a general graph, we cannot infer the likelihood exactly. However nested learning and inference can still be applied using approximate inference algorithms such as convex sum-product belief propagation (Wainwright et al., 2005; Heskes, 2006; Meltzer et al., 2009; Hazan and Shashua, 2010). Nevertheless, the approximate inference algorithms might be computationally expensive to be used as a nested subroutine of the learning algorithm.

In this article we suggest to interleave optimization of learning parameters with optimization of inference parameters. We call this approach blending learning and inference in CRFs. For this end we propose an optimization program that maximizes log-beliefs, i.e., probability distributions over subsets of training variables that agree on their marginal probabilities. These beliefs are elements of the local polytope (Wainwright and Jordan, 2008). The log-beliefs objective is concave and consists of two types of variables that are related to the learning and inference tasks respectively. We are able to blend the learning and inference procedures by alternating maximizations over the inference and learning variables. With blending we reach the nested learning-inference optimum much faster.

We also define loss-adjusted beliefs to integrate prior knowledge about the desired inference as well as a parameter that controls the smoothness of the beliefs. In the past and partly due to its efficiency, the presented machine learning algorithm was shown to improve the state-of-the-art in various computer vision tasks, including 2D scene understanding (Yao et al., 2012), 3D scene understanding (Lin et al., 2013), shape reconstruction (Salzmann and Urtasun, 2012), indoor scene understanding (Schwing et al., 2012a; Schwing and Urtasun, 2012), depth estimation (Yamaguchi et al., 2012), flow estimation (Yamaguchi et al., 2013) and visual-language understanding (Fidler et al., 2013). This manuscript extends our previous work (Hazan and Urtasun, 2010) to high order setting while simplifying its theory and proofs. The code is publicly available on `http://www.alexander-schwing.de/projectsGeneralStructuredPredictionLatentVariables.php`.

The reminder of the paper is organized as follows. In Section 3 we review the parameter learning setting of CRFs that maximize the log-likelihood of training labels given corresponding data instances. We also present the nested learning and inference approach as well as approximate inference algorithms of belief propagation and its convex variants. In Section 4 we describe the objective function for nested learning and inference that maximize the log-beliefs. We then describe a blending algorithm to optimize this objective and describe its convergence properties using convex duality. Next we present loss-adjusted beliefs and the appropriate modifications for blending in Section 5, drawing connections to blending in the structured SVMs setting suggested by Meshi et al. (2010). We then demonstrate the effectiveness of our approach in Section 6. We conclude by describing the generality of our approach, relating blending and convexity to the penalty method.

2. Related work

Learning log-beliefs extends the CRFs framework that maximizes the log-likelihood of conditional Gibbs distributions (cf. Lafferty et al. (2001); Lebanon and Lafferty (2002)). Gibbs distributions, also known as Markov random fields, are probability distributions that are defined on a product space using potential functions over subsets of variables. Gibbs probability models often consider exponentially many possible assignments for these variables. In this case, approximate inference methods are used in a black box manner to estimate the gradient and the objective of CRFs, resulting in the nested learning-inference algorithm illustrated in Figure 1. Nested learning-inference algorithms are successfully dealing with real-world problems, e.g., (Levin and Weiss, 2006) in computer vision, (Yanover et al., 2007) in computational biology and (Sutton and McCallum, 2009) in language processing to name a few.

The current work extends and simplifies our previous work (Hazan and Urtasun, 2010). We simplify the learning objective while formulating it as the maximization of log-beliefs, which are pseudo-marginal probabilities of the Gibbs distribution. We extend the learning procedure while considering pseudo-marginals of Gibbs distributions on any subset of its variables. In the last couple of years, the presented machine learning algorithm was shown to improve the state-of-the-art in various computer vision tasks: Considering outdoor scene understanding (Yao et al., 2012), each (super)pixel is represented by a variable of the Gibbs model and its possible assignments correspond to discrete semantic label, e.g., person, car, tree and so on. Our learning algorithm estimates the parameters that maximize the probability of the training data per variable (i.e., pixel and its observed object) and subsets of variables (e.g., neighborhood of pixels and their observed objects). Considering indoor scene understanding (Schwing et al., 2012a; Schwing and Urtasun, 2012; Lin et al., 2013) each wall or a 3D object in the room is represented by a subsets of variables and our learning algorithm estimates the position of these objects while maximizing likelihood of these subsets within the training data. In depth estimation and optimal flow (Yamaguchi et al., 2012, 2013) the variables of the Gibbs models are either continuous or discrete. The continuous variables correspond to the possible hyperplanes that either explain the depth or the optical flow of the super pixels in the training images. The discrete variables maintain consistency between adjacent super pixels. Our learning algorithm estimates the probable hyperplanes and their spatial relations in the training data.

Our approach suggests to blend the inference and learning steps and reach the same optimum as nested approaches while being at least an order of magnitude faster. Blending helps the algorithm to avoid computationally expensive inference algorithms when learning parameters w that are far from optimal. Similar observations have been made in the context of coordinate descent by Tappenden et al. (2013). Other approaches for blending CRFs appear are developed by Domke (2011). Lemma 3 describes how to infer non-consistent beliefs. These beliefs are different from the beliefs that are usually derived during the runtime of approximate inference algorithms. The theoretical characterization of the optimal points for the learning-inference procedures are described by Wainwright (2006); Wainwright et al. (2003).

Meshi et al. (2010) describe blending learning and inference in the context of structured SVMs, which are constructed to minimize the loss between the predicted labels and the

observed ones (cf. (Taskar et al., 2004; Tsochantaridis et al., 2004; Collins, 2002)). The ideas of blending learning and inference in structured SVMs also appear in (Taskar et al., 2005; Anguelov et al., 2005). In contrast, our work focuses on blending learning and inference when maximizing log-probabilities. Nevertheless, our loss-adjusted beliefs may describe a probabilistic alternative to blended learning in structured-SVMs. When setting the counting numbers $c_r = 0$, we effectively work with zero-one probabilities (i.e., max-beliefs) thus we recover the algorithm of Meshi et al. (2010).

3. Background

Log-likelihood learning in structured models involves data instances $x \in \mathcal{X}$ and their labels $y \in \mathcal{Y}$. The structure is incorporated into the labels which may refer to sequences, grids, or other high-dimensional objects. For every data instance x, its possible labels are described by a set of feature functions $\phi_k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, $k \in \{1, \dots, K\}$. A linear combination of the K features is used to score the different labels $y \in \mathcal{Y}$ using the parameters $w \in \mathbb{R}^K$. Formally we obtain the score $\theta(y; x, w)$ of a label via

$$\theta(y; x, w) = \sum_k w_k \phi_k(x, y).$$

The real valued score is mapped to the probability scale via the Gibbs distribution:

$$p(y|x; w) \propto \exp(\theta(y; x, w)). \tag{1}$$

Within the CRF framework, the goal is to learn the parameters of the potential functions to maximize the conditional likelihood of the training data $(x, y) \in \mathcal{S}$:

$$\max_w \sum_{(x,y)\in\mathcal{S}} \log p(y|x; w) - \frac{C}{2}\|w\|_2^2. \tag{2}$$

The regularization term is sometimes considered as a Gaussian prior over the parameters w. The regularized likelihood of CRFs is a concave and smooth function and its optimal parameters may be attained by gradient ascent. The gradient measures the disagreements between the inferred distribution over labels and the groundtruth training labels, i.e.,

$$\frac{\partial \ \log p(y|x; w)}{\partial w_k} = \sum_{\hat{y}\in\mathcal{Y}} p(\hat{y}|x; w)\phi_k(x, \hat{y}) - \phi_k(x, y). \tag{3}$$

The computational complexity of CRFs is governed by inference, which amounts to evaluating the probability $p(\hat{y}|x; w)$ for computing the objective and the gradient.

We consider cases in which the labels $y \in \mathcal{Y}$ are n-tuples, i.e., $y = (y_1, ..., y_n)$, and hence the configuration space is exponential in n. The features describe relations between subsets of elements $r \subset \{1, ..., n\}$, also called regions or factors. We denote by $\mathcal{R}_k$ the regions required to compute the feature $\phi_k(x, y)$. Importantly, the features are functions of their regions labels $y_r \subset \{y_1, ..., y_n\}$, i.e.,

$$\phi_k(x, y_1, ..., y_n) = \sum_{r\in\mathcal{R}_k} \phi_{k,r}(x, y_r). \tag{4}$$

Thus the features define hypergraphs whose nodes represent the n labels indexes, and the regions $\mathcal{R} = \cup_k \mathcal{R}_k$ correspond to its hyperedges. A convenient way to represent a hypergraph is by its region graph. A region graph is a directed graph whose nodes represent the regions and its direct edges correspond to the inclusion relation, i.e., a directed edge from node r to s is possible only if $s \subset r$. We adopt the terminology where the sets $P(r)$ and $C(r)$ represent all nodes that are parents and children of the node r, respectively.

The Hammersley-Clifford theorem (e.g., Lauritzen (1996)) asserts that the Gibbs distributions $p(\hat{y}|x;w)$ defined in Equation (1) correspond to a Markov random field (MRF) whose statistical independencies are described by the joint hypergraph. These independencies are determined by the Markov property: two nodes in the graph are conditionally independent when they are separated by observed nodes. Aji and McEliece (2001) show that whenever the region graph is bipartite and has no cycles, the Markov property provides a low dimensional representation of the Gibbs distribution using its marginal probabilities $p(\hat{y}_r|x;w) = \sum_{\hat{y}\setminus\hat{y}_r} p(\hat{y}|x;w)$, namely

$$p(\hat{y}|x;w) = \prod_{r\in\mathcal{R}} p(\hat{y}_r|x;w)^{1-|P(r)|}. \tag{5}$$

In such cases the inference step, i.e., estimating the probabilities $p(\hat{y}|x;w)$ can be performed efficiently using message-passing algorithms. When the region graph has no cycles it is bipartite, therefore it has two types of regions: outer regions, i.e., regions that are not contained by other regions, and inner regions. To differentiate between those regions we denote outer regions by α and inner regions by i. In this case, one can use the belief propagation algorithm to efficiently infer the marginal probabilities without performing exponentially many operations:

Algorithm 1 Sum-product belief propagation

Set $\mathcal{K}_r = \{k : r \in \mathcal{R}_k\}$. For every $(x,y), w$ set $\theta_r(\hat{y}_r) = \sum_{k\in\mathcal{K}_r} w_k \phi_{k,r}(x,\hat{y}_r)$.
Repeat until convergence:

$$\mu_{\alpha\to i}(y_i) = \log\Big(\sum_{y_\alpha\setminus y_i} \exp\big((\theta_\alpha(y_\alpha) + \sum_{j\in C(\alpha)\setminus i} \lambda_{j\to\alpha}(y_j))\big)\Big)$$

$$\lambda_{i\to\alpha}(y_i) = \theta_i(y_i) + \sum_{\beta\in P(i)\setminus\alpha} \mu_{\beta\to i}(y_i)$$

Output:

$$b_i(y_i) \propto \exp\big(\theta_i(y_i) + \sum_{\alpha\in P(i)} \mu_{\alpha\to i}(y_i)\big)$$

$$b_\alpha(y_\alpha) \propto \big(\theta_\alpha(y_\alpha) + \sum_{i\in C(\alpha)} \lambda_{i\to\alpha}(y_i)\big)$$

The marginal probabilities $p(\hat{y}_r|x;w)$ appear as the beliefs $b_r(\hat{y}_r)$. In general, when the region graph has cycles, the belief propagation algorithm is no longer guaranteed to output the marginal probabilities. Nevertheless, when it converges it provides beliefs that agree on their marginal probabilities, namely $\sum_{y_\alpha\setminus y_i} b_\alpha(y_\alpha) = b_i(y_i)$. In some cases the belief propagation algorithm infers beliefs $b_r(\hat{y}_r)$ which approximate well the marginal probabilities, while in other cases it produces non-accurate results or might even fail to converge. A possi-

ble explanation for this behavior comes from the fact that the belief propagation algorithm iterates the stationary points of a non-convex objective called the Bethe free energy (Yedidia et al., 2005). Recently, in an extensive effort to derive converging belief propagation type algorithms, the non-convex Bethe free energy was replaced by convex free energies while introducing nonnegative counting numbers c_r that replace the Bethe coefficients. Consequently, the belief propagation algorithm was replaced by block coordinate descent over the dual program (Heskes, 2006; Meltzer et al., 2009; Hazan and Shashua, 2008). These dual block coordinate descent algorithms belong to the family of the norm-product belief propagation algorithms:

Algorithm 2 Norm-Product Belief Propagation

Set $\hat{c}_i = c_i + \sum_{\alpha \in P(i)} c_\alpha$. Repeat until convergence:

$$\mu_{\alpha \to i}(y_i) = c_\alpha \log \Big(\sum_{y_\alpha \setminus y_i} \exp \big((\theta_\alpha(y_\alpha) + \sum_{j \in C(\alpha) \setminus i} \lambda_{j \to \alpha}(y_j)) / c_\alpha \big) \Big)$$

$$\lambda_{i \to \alpha}(y_i) = \tfrac{c_\alpha}{\hat{c}_i} \Big(\theta_i(y_i) + \sum_{\beta \in P(i)} \mu_{\beta \to i}(y_i) \Big) - \mu_{\alpha \to i}(y_i)$$

Output:

$$b_i(y_i) \propto \exp \big(\theta_i(y_i) + \sum_{\alpha \in P(i)} \mu_{\alpha \to i}(y_i) \big)^{1/\hat{c}_i}$$

$$b_\alpha(y_\alpha) \propto \exp \big(\theta_\alpha(y_\alpha) + \sum_{i \in C(\alpha)} \lambda_{i \to \alpha}(y_i) \big)^{1/c_\alpha}$$

The norm-product algorithm, illustrated in Algorithm 2, reduces to belief propagation when setting its coefficients to $c_r = 1 - |P(r)|$, namely, the Bethe counting numbers. We refer the interested reader to (Wainwright and Jordan, 2008) for more details.

The norm-product algorithm iterates over the fixed point solutions for the variational problem

$$\arg \max_{b \in L(G)} \sum_{r, y_r} b_r(y_r) \theta_r(y_r) + \sum_r c_r H(b_r). \tag{6}$$

The set $L(G)$ is known as the local polytope, and contains probability distributions $b_r(y_r)$ that agree on their overlapping variables, i.e.,

$$L(G) = \Big\{ b_r(y_r) \ : \ b_r(y_r) \geq 0, \sum_{y_r} b_r(y_r) = 1, \ \forall p \in P(r) \sum_{y_p \setminus y_r} b_p(y_p) = b_r(y_r) \Big\}. \tag{7}$$

Throughout this work we refer to elements in the local polytope as beliefs.

The variational program given in Equation (6) is concave whenever $c_r \geq 0$. The norm-product algorithm given in Algorithm 2 performs block coordinate descent on its dual program. Therefore it is guaranteed to converge to beliefs that agree on their marginal probabilities. Typically its inferred beliefs approximate the marginal probabilities as well as the belief propagation approximations (Meshi et al., 2009). Thus in its various forms it is used as the inference step when learning the parameters of CRFs. Such nested loop algorithms for performing learning and inference are presented in Figure 1. Unfortunately, iteratively executing the norm-product algorithm as an inference procedure to compute

Learning with nested inference for CRFs

1. Set $\theta_r(y_r; x, w) = \sum_{k \in \mathcal{K}_r} w_k \phi_{k,r}(x, y_r)$. Repeat until convergence:
2. Inference: For every $(x, y) \in \mathcal{S}$:

$$b^* = \arg \max_{b(\cdot|x) \in L(G)} \sum_{r \in \mathcal{R}, y_r \in \mathcal{Y}_r} b_r(y_r|x) \theta_r(y_r; x, w) + \sum_{r \in \mathcal{R}} H(b_r).$$

3. Learning:

$$w_k \leftarrow w_k - \eta \Big(\sum_{(x,y) \in \mathcal{S}} \sum_{r \in \mathcal{R}} \Big(\sum_{\hat{y}_r} b_r^*(\hat{y}_r|x) \phi_{k,r}(x, \hat{y}_r) - \phi_{k,r}(x, y_r) \Big) + C w_k \Big).$$

Figure 1: Nested (unblended) inference in learning. The inference step is performed in every iteration. The learning step improves the parameters till learning is attained. η is typically referred as the learning rate and may be set as $1/\sqrt{t}$, where t is the iteration index. The abbreviation $b \in L(G)$ describes conditional beliefs $b(\cdot|x)$ for every $x \in S$. Each of these conditional beliefs is in the local polytope, as defined in Equation (7).

the gradient is computationally demanding and this method has not been used widely (see Section 2 for more details). In the following we explore duality to provide the means to blend the learning and inference tasks efficiently.

4. Blending learning and inference

Log-likelihood of Gibbs distributions, as defined in Equation (1) with potentials $\theta(y; x, w)$ that are linear functions of their parameters w, results in a concave program. When using nested (unblended) inference, the learning algorithm executes a concave program for inferring beliefs about its marginal probabilities that are required for computing its gradient. Our main result explicitly defines the concave program whose optimal solutions are the limit points of the nested learning and inference algorithm that is shown in Figure 1. Using this characterization we are able to derive an algorithm that blends the learning and inference steps. Consequently it is orders of magnitude faster than the nested algorithm that uses inference as a black-box algorithm. Since our blended algorithm optimizes a concave program, it is guaranteed to reach the same optimum as the nested algorithm.

Theorem 1 *The limit points of the nested inference and learning algorithm in Figure 1 are described by the optimal points of the following concave program maximizing log-beliefs:*

$$\max_{w,\lambda} \sum_{(x,y) \in S} \sum_{r \in \mathcal{R}} \log b_r(y_r|x; w, \lambda) - \frac{C}{2} \|w\|^2$$

$$s.t \quad b_r(y_r|x; w, \lambda) \propto \exp \Big(\theta_r(y_r; x, w) + \sum_{c \in C(r)} \lambda_{c \to r}(y_c; x) - \sum_{p \in P(r)} \lambda_{r \to p}(y_r; x) \Big)$$

Proof The theorem replaces maximization over $b \in L(G)$ with maximization over λ. Decoupling the maximizations takes the form $\max_w \sum_{(x,y)\in S} \big(\max_\lambda \{ \sum_{r\in\mathcal{R}} \log b_r(y_r|x;w,\lambda) \} \big)$. In the following we show how to derive the result from Lagrange optimality conditions (KKT):

$$\begin{aligned}
\lambda^* &= \arg\max_\lambda \sum_{r\in\mathcal{R}} \log b_r(y_r|x;w,\lambda) \\
b^* &= \arg\max_{b\in L(G)} \sum_{r\in\mathcal{R}, y_r\in\mathcal{Y}_r} b_r(y_r|x)\theta_r(y_r;x,w) + \sum_{r\in\mathcal{R}} H(b_r) \\
&\Longrightarrow b_r^*(y_r|x) \propto \exp\Big(\theta_r(y_r;x,w) + \sum_{c\in C(r)} \lambda^*_{c\to r}(y_c;x) - \sum_{p\in P(r)} \lambda^*_{r\to p}(y_r;x)\Big) \qquad (8)
\end{aligned}$$

Using the above Lagrange optimality conditions, the theorem follows since the inference nested within the learning algorithm applies gradient ascent to the following program:

$$\max_w \sum_{(x,y)\in S} \big(\max_\lambda \{ \sum_{r\in\mathcal{R}} \log b_r(y_r|x;w,\lambda) \} \big) - \frac{C}{2}\|w\|^2 = \max_w \sum_{(x,y)\in S} \sum_{r\in\mathcal{R}} \log b_r^*(y_r|x) - \frac{C}{2}\|w\|^2.$$

Equation (8) states the Lagrange optimality conditions for maximum-likelihood maximum-entropy type duality. Consider the constrained inference algorithm in Figure (1) and the Lagrange multipliers $\lambda_{r\to p}(y_r;x,w)$ for the marginalization constraints $\sum_{y_p\setminus y_r} b_p(y_p|x) = b_r(y_r|x)$. Its corresponding Lagrangian is $L(b,\lambda) = \sum_{r,y_r} b_r(y_r|x)\theta_r(y_r;x,w,\lambda) + \sum_r H(b_r)$ where $\theta_r(y_r;x,w,\lambda) = \theta_r(y_r;x,w) + \sum_{c\in C(r)} \lambda_{c\to r}(y_c;x) - \sum_{p\in P(r)} \lambda_{r\to p}(y_r;x)$. Its dual function is $q(\lambda) = \max_b L(b,\lambda)$ while $b_r(y_r|x)$ are subject to probability constraints. Conjugate duality between the entropy function and the log-partition function (cf. Wainwright and Jordan (2008)) implies that $q(\lambda) = \sum_r \log(\sum_{y_r} \exp(\theta_r(y_r;x,w,\lambda)))$. Since strong duality between the entropy and the log-partition function holds, its Lagrange optimality conditions imply

$$\lambda^* = \arg\min_\lambda \sum_{r\in\mathcal{R}} \log\Big(\sum_{y_r} \exp\big(\theta_r(y_r;x,w,\lambda)\big)\Big).$$

The theorem then follows since $\sum_{r\in\mathcal{R}} \sum_{p\in P(r)} \lambda_{r\to p}(y_r;x) - \sum_{r\in\mathcal{R}} \sum_{c\in C(r)} \lambda_{c\to r}(y_c;x) \equiv 0$ therefore $\sum_r \log b_r^*(y_r|x) = -q(\lambda^*)$ and the Lagrange optimality conditions in Equation (8) hold. Thus $b_r^*(y_r|x)$ can be replaced by its re-parametrization $\frac{1}{Z_r}\exp(\theta_r(y_r;x,w,\lambda^*))$, with $Z_r = \sum_{\hat{y}_r} \exp(\theta_r(\hat{y}_r;x,w,\lambda^*))$.

■

The maximum log-beliefs program describes the variational landscape of nested inference in learning. Thus it can be used to measure the step-size η in Figure 1. For example, the Armijo rule that determines the step size according to the variational neighborhood results in a faster convergence per iteration than using a fixed learning rate.

The nested learning and inference algorithm performs a complete inference step before performing a single learning step. The inference step derives beliefs that agree on their marginal probabilities. In this work we use the maximum log-beliefs concave program to blend the learning and inference steps. Specifically, we use block coordinate ascent steps

to blend learning (optimization w.r.t. w) with incomplete inference steps (optimization w.r.t. λ) that derive beliefs $b_r(y_r|x;w,\lambda)$ that not necessarily agree on their marginal probabilities, i.e., $\sum_{y_p \setminus y_r} b_p(y_p|x;w,\lambda) = b_r(y_r|x;w,\lambda)$. Such an approach is computationally favorable since it does not require to perform a complete inference step for initial learning parameters. Concavity ensures that blending reaches the maximum log-beliefs optimum thus it guarantees to derive consistent beliefs upon convergence.

Performing block coordinate descent on the maximum log-beliefs program in Theorem 1 requires minimizing a block of variables while holding the rest fixed. We begin by describing how to infer the optimal set of variables $\lambda_{r\to p}(y_r;x)$ that are related to a region and its parents in the graphical model.

Lemma 2 Blended inference: *Consider the program given in Theorem 1. For a given region r, the optimal inference parameters $\lambda^*_{r\to p}(y_r;x)$, for every $p \in P(r), y_r \in \mathcal{Y}_r, x \in \mathcal{S}$, when fixing all other λ takes the following form:*

$$\begin{aligned}
\mu_{p\to r}(y_r;x) &= \log\Big(\sum_{y_p\setminus y_r} \exp\big(\theta_p(y_p;x,w) + \sum_{c\in C(p)\setminus r} \lambda_{c\to p}(y_c;x) - \sum_{p'\in P(p)} \lambda_{p\to p'}(\hat{y}_p;x)\big)\Big) \\
\lambda^*_{r\to p}(y_r;x) &= \frac{\theta_r(y_r;x,w) + \sum_{c\in C(r)} \lambda_{c\to r}(y_c;x) + \sum_{p'\in P(r)} \mu_{p'\to r}(y_r;x)}{1+|P(r)|} - \mu_{p\to r}(y_r;x)
\end{aligned}$$

Proof The maximum log-beliefs program in Theorem 1 is smooth, concave and unconstrained as a function of λ, therefore the optimum is achieved when the gradient vanishes. Setting

$$\theta_r(y_r;x,w,\lambda) = \theta_r(y_r;x,w) + \sum_{c\in C(r)} \lambda_{c\to r}(y_c;x) - \sum_{p\in P(r)} \lambda_{r\to p}(y_r;x) \tag{9}$$

and $b_r(y_r|x;w,\lambda) \propto \exp(\theta_r(y_r;x,w,\lambda))$, the gradient with respect to $\lambda_{r\to p}(y_p;x)$ takes the form

$$\frac{\partial \sum_{(x,y),r} \log b_r(y_r|x;w,\lambda)}{\partial \lambda_{r\to p}(y_r;x)} = \sum_{y_p\setminus y_r} b_p(y_p|x;w,\lambda) - b_r(y_r|x;w,\lambda).$$

The optimal dual variables are those for which the gradient vanishes, i.e., the corresponding beliefs agree on their marginal probabilities. When setting $\mu_{p\to r}(y_r;x)$ as above, the marginalization of $b_p(y_p|x;w,\lambda)$ satisfies

$$\sum_{y_p\setminus y_r} b_p(\hat{y}_p|x;w,\lambda) \propto \exp\big(\mu_{p\to r}(y_r;x) + \lambda_{r\to p}(y_r;x)\big).$$

Therefore, by taking the logarithm, the gradient vanishes whenever the beliefs numerators agree up to an additive constant:

$$\sum_{y_p\setminus y_r} b_p(y_p|x;w,\lambda^*) - b_r(y_r|x;w,\lambda^*) = 0 \Longleftrightarrow \mu_{p\to r}(y_r;x) + \lambda^*_{r\to p}(y_r;x) = \theta_r(\hat{y}_r;x,w,\lambda^*).$$

The right hand side of the condition almost characterizes completely the optimal variables $\lambda^*_{r\to p}(y_r;x)$ by $\lambda^*_{r\to p}(y_r;x) = \theta_r(\hat{y}_r;x,w,\lambda^*) - \mu_{p\to r}(y_r;x)$. Unfortunately, $\theta_r(\hat{y}_r;x,w,\lambda^*)$ depends on $\sum_{p\in P(r)} \lambda^*_{r\to p}(y_r;x)$ thus it cannot serve as an update rule in its current form.

To complete the proof we require to replace $\sum_{p\in P(r)} \lambda^*_{r\to p}(y_r;x)$ with another quantity that does not depend on $\lambda^*_{r\to p}(y_r;x)$ for every $p \in P(r)$ and $y_r \in \mathcal{Y}_r$.

To isolate this quantity we sum both sides of the equality $\mu_{p\to r}(y_r;x) + \lambda^*_{r\to p}(y_r;x) = \theta_r(\hat{y}_r;x,w,\lambda^*)$ with respect to $p \in P(r)$, thus we are able to obtain

$$(1+|P(r)|)\sum_{p\in P(r)} \lambda^*_{r\to p}(y_r;x) = |P(r)|(\theta_r(y_r;x,w) + \sum_{c\in C(r)} \lambda_{c\to r}(y_c;x)) - \sum_{p\in P(r)} \mu_{p\to r}(y_r;x)$$

Plugging it into the above equation results in the desired block dual ascent update rule. ■

One can verify that since the program is not strictly concave in λ, the optimal solutions can be achieved for every additive shift of $\lambda_{r\to p}(y_r;x)$. The above lemma describes an analytic solution for the optimal $\lambda_{r\to p}(y_r;x)$, that are computed in the block coordinate steps of the algorithm. In practice, block coordinate descent with analytic steps provides a significant speedup over conventional gradient methods and can be parallelized and distributed easily, as shown by Schwing et al. (2011).

A learning step updates the weights w so as to maximize the log-beliefs. When using blended inference, the beliefs are not required to agree on their marginal probabilities. However, they are governed by the concave program in Theorem 1. Its concavity guarantees that these beliefs agree on their marginals at the optimum.

Lemma 3 Blended learning: *Consider the program given in Theorem 1. The gradient of its objective function with respect to w_k takes the form:*

$$\sum_{(x,y)\in\mathcal{S}} \sum_{r\in\mathcal{R}_k} \Big(\sum_{\hat{y}_r\in\mathcal{Y}_r} b_r(\hat{y}_r|x;w,\lambda)\phi_{k,r}(x,\hat{y}_r) - \phi_{k,r}(x,y_r)\Big) + Cw_k.$$

Proof We let

$$\log b_r(y_r|x;w,\lambda) = \theta_r(y_r;x,w,\lambda) - \log\Big(\sum_{\hat{y}_r} \exp\big(\theta_r(\hat{y}_r;x,w,\lambda)\big)\Big).$$

The theorem follows by noting that $\theta_r(y_r;x,w) = \sum_{k\in\mathcal{K}_r} w_k\phi_{k,r}(x,y_r)$, recalling the definition of $\theta_r(y_r;x,w,\lambda)$ in Equation (9) and that $b_r(\hat{y}_r|x;w,\lambda)$ which is defined in Theorem 1 is the gradient of its log-partition function, i.e., $\log\Big(\sum_{\hat{y}_r} \exp\big(\theta_r(\hat{y}_r;x,w,\lambda)\big)\Big)$. ■

The computational complexity of the gradient depends on the structure of the features, especially the number of regions and their labels. Our framework prefers features with small regions and reasonable number of labels. Another computational issue relates to the step size η for increasing the objective along the gradient of w_k. In general, the gradient updates verify that the chosen step size η reduces the objective. Theoretically, we can use the fact that the gradient is Lipschitz continuous to predetermine a step size that guarantees ascent. In practice it gives worse performance than searching for a step size depending on the gradient and the objective at any given point.

Lemmas 2 and 3 describe the inference and learning steps for maximizing the log-beliefs maximization in Theorem 1. Since the program is concave, the order of the maximization steps is not important, and as long as all inference and learning parameters are optimized the maximal value is attained. For example, one can maximize the inference variables λ till

Blending learning and inference

1. Set $\theta_r(y_r; x, w) = \sum_{k \in \mathcal{K}_r} w_k \phi_{k,r}(x, y_r)$. Repeat until convergence:
2. For every $(x, y) \in \mathcal{S}$, $r \in \mathcal{R}$, $\hat{y}_r \in \mathcal{Y}_r$, $p \in P(r)$:

$$\mu_{p \to r}(y_r; x) = \log \Big(\sum_{y_p \setminus y_r} \exp \big(\theta_p(y_p; x, w) + \sum_{c \in C(p) \setminus r} \lambda_{c \to p}(y_c; x) - \sum_{p' \in P(p)} \lambda_{p \to p'}(\hat{y}_p; x) \big) \Big)$$

$$\lambda_{r \to p}(y_r; x) = \frac{\theta_r(y_r; x, w) + \sum_{c \in C(r)} \lambda_{c \to r}(y_c; x) + \sum_{p' \in P(r)} \mu_{p' \to r}(y_r; x)}{1 + |P(r)|} - \mu_{p \to r}(y_r; x)$$

3. Set $b_r(y_r | x; w, \lambda) \propto \exp \big(\theta_r(y_r; x, w) + \sum_{c \in C(r)} \lambda_{c \to r}(y_c; x) - \sum_{p \in P(r)} \lambda_{r \to p}(y_r; x) \big)$.

$$w_k \leftarrow w_k - \eta \Big(\sum_{(x,y) \in \mathcal{S}} \sum_{r \in \mathcal{R}_k} \Big(\sum_{\hat{y}_r \in \mathcal{Y}_r} b_r(\hat{y}_r | x; w, \lambda) \phi_{k,r}(x, \hat{y}_r) - \phi_{k,r}(x, y_r) \Big) + C w_k \Big).$$

Figure 2: The inference step is described in Lemma 2 and the learning step is described in Lemma 3. The step size η is set to guarantee convergence (e.g., corresponding to the Lipschitz constant or the Armijo rule.) Concavity of the program in Theorem 1 ensures that the blending converges to consistent inferred beliefs, see Theorem 4.

they do not change before optimizing the learning parameters w. We refer to this approach in Figure 1 as nested inference within learning since it performs the approximate inference heuristic described in Section 3 as a black-box solver. Nested learning and inference is computationally unfavorable in general as it requires to infer λ till convergence for every gradient step for learning w. Since concavity ensures that the maximization does not depend on the order of the maximizing steps, it also provides a principled way to blend the learning and inference steps. Particularly, it may learn the w parameters using inferred beliefs $b_r(y_r | x; w, \lambda) \propto \exp(\theta_r(y_r, x, w, \lambda))$ that do not agree on their marginal probabilities. For this purpose our algorithm infers the parametrized beliefs *differently* than the (outer) beliefs that are computed by the nested learning algorithm in Figure 1. This blending property is important in practice, since shortly upon initialization, where the given parameters w are far from the optimum, it is not advisable to spend time on computing consistent beliefs. Figure 2 summarizes the inference-learning blending algorithm.

The block coordinate descent algorithm is guaranteed to converge, as it monotonically increases the log-beliefs in Theorem 1, which are upper bounded by its dual. Moreover, the values that are generated by the algorithm converge to the program's optimal value (see Theorem 4 for exact statement). It is not immediately clear that the algorithm converges to its optimal value since the program is not strictly concave. Consequently, the sequence of variables $\lambda_{r \to p}(y_r; x)$ generated by the algorithm is not guaranteed to be bounded. As a trivial example, adding an arbitrary constant to the variables, $\lambda_{r \to p}(y_r; x) + c$, does not

change the objective value, hence the algorithm can generate a monotonically decreasing unbounded sequences. Convergence to the optimum holds by convex duality:

Theorem 4 *The learning-inference blending algorithm in Figure 2 for log-beliefs maximization is guaranteed to converge. Moreover, the value of its objective is guaranteed to converge to the global maximum, and its sequence of beliefs are guaranteed to converge to consistent beliefs that are the unique solution of the dual program*

$$\begin{aligned}
&\min_{z,b_r} \sum_{(x,y)\in\mathcal{S}} \frac{1}{2C}\|z\|^2 - \sum_{r\in\mathcal{R}} H(b_r) \\
&\text{subject to } \; b_r(\hat{y}_r|x) \geq 0, \quad \sum_{\hat{y}_r} b_r(\hat{y}_r|x) = 1, \quad b_r(\hat{y}_r|x) = \sum_{\hat{y}_p \setminus \hat{y}_r} b_p(\hat{y}_p|x) \\
&\qquad\qquad z_k = \sum_{(x,y)\in\mathcal{S}} \sum_{r\in\mathcal{R}_k} \Big(\sum_{\hat{y}_r} b_r(\hat{y}_r|x)\phi_{k,r}(x,\hat{y}_r) - \phi_{k,r}(x,y_r) \Big)
\end{aligned}$$

Proof The update rules in Figure 2 iteratively apply the block coordinate ascent rules in Lemmas 2 and 3 thus monotonically increase the primal objective in Theorem 1. This program is concave thus it is bounded by its dual program, therefore the value of its objective is guaranteed to converge. To derive the dual program we construct the Lagrangian

$$\begin{aligned}
L(w,\lambda,b) \;\; = \;\; & -\frac{C}{2}\|w\|^2 + \sum_k w_k \Big(\sum_{(x,y)\in S, r\in\mathcal{R}_k} \phi_{k,r}(x,y_r) \Big) \\
& - \sum_{(x,y),r} \log\Big(\sum_{\hat{y}_r} \exp(\theta_r(\hat{y}_r;x) + \sum_{c\in C(r)} \lambda_{c\to r}(\hat{y}_c;x) - \sum_{p\in P(r)} \lambda_{r\to p}(\hat{y}_r;x)) \Big) \\
& + \sum_{(x,y),r} \sum_{\hat{y}_r} b_r(\hat{y}_r|x) \Big(\theta_r(\hat{y}_r;x) - \sum_k w_k \phi_{k,r}(x,\hat{y}_r) \Big).
\end{aligned}$$

The variables $b_r(\hat{y}_r|x)$ are the Lagrange multipliers for the equality constraints $\theta_r(\hat{y}_r;x) = \sum_k w_k \phi_{k,r}(x,y_k)$. The dual program takes the form $q(b) = \max_{w,\lambda} L(w,\lambda,b)$. Setting z_k as above, since the $\|\cdot\|^2$ is the conjugate dual of $\|\cdot\|^2$, the maximization over w takes the form $\max_w\{-\frac{C}{2}\|w\|^2 - \sum_k w_k z_k\} = \frac{1}{2C}\|z\|^2$. To complete the derivation of the dual, the maximization over λ, for every (x,y) takes the form $\max_\lambda\{\sum_{r,\hat{y}_r} b_r(\hat{y}_r|x)(\theta_r(\hat{y}_r;x) - \sum_r \log(\sum_{\hat{y}_r} \exp(\theta_r(\hat{y}_r;x) + \sum_{c\in C(r)} \lambda_{c\to r}(\hat{y}_c;x) - \sum_{p\in P(r)} \lambda_{r\to p}(\hat{y}_r;x)))\}$. This is the conjugate dual of the log-partition function, which is known to be the entropy function $H(b_r)$. The mixing of the messages between the different regions results in the marginalization constraints in the dual program. An alternative proof for the conjugate duality between the re-parametrized log-partitions and entropies subject to marginalization constraints appears in the more generalized setting of Theorem 5.

Finally, since the dual is strictly convex subject to linear marginalization constraints and the linear moment constraints the convergence properties are a consequence of Tseng and Bertsekas (1987). ■

The convergence of the block coordinate ascent depends on the step size η, which requires to increase the log-beliefs. This can be done by the Armijo rule, or by using the fact that

the function z_k^2 is strongly convex (e.g., Tseng and Bertsekas (1987)) and its gradient is Lipschitz continuous (e.g., Nesterov (2004)). In practice, Theorem 4 describes how to measure the convergence of the algorithm. Specifically, it may be derived from the primal objective value, the dual objective value or the beliefs themselves, as all these quantities converge. Unfortunately, the variables λ might not converge, but if they do converge, their convergence point is optimal.

5. Blending learning and loss adjusted inference

Loss adjusted inference emerges from Support Vector Machines (SVMs) where a loss function indicates preferences between different labels. In the following we consider nonnegative loss functions $\ell_r(y_r, \hat{y}_r) \geq 0$ over subsets of variables, while $\ell_r(y_r, y_r) = 0$. We suggest to augment our learned beliefs with loss adjusted probability models. Given a training example (x, y), we define the loss adjusted belief model to be

$$b_r(\hat{y}_r|x; w, \lambda) \propto \exp\Big(\ell_r(y_r, \hat{y}_r) + \theta_r(\hat{y}_r; x, w, \lambda)\Big).$$

Note that these beliefs should be conditioned over x as well as the vector $\ell_r(\cdot, y_r)$. However, to simplify the notation, we leave this conditioning implicit. The intuition for using the loss as a prior and for deriving loss adjusted beliefs is based on encouraging to learn parameters that decrease probabilities over labels with higher loss with respect to the training labels. The likelihood approach aims at maximizing the beliefs $b_r(y_r|x; w, \lambda)$. Since the beliefs are probability distributions, it equivalently aims at minimizing the log-beliefs of all other assignments, namely, $b_r(\hat{y}_r|x; w, \lambda)$ for every $\hat{y}_r \neq y_r$. Since the loss $\ell_r(y_r, \hat{y}_r)$ is a nonnegative function it implies that loss-adjusted maximum-likelihood learns parameters that better reduce scores of non-observed training labels, namely $\theta_r(\hat{y}_r; x, w, \lambda)$ for any $\hat{y}_r \neq y_r$. The algorithms for maximizing loss adjusted beliefs models follow the derivations presented in Section 4, while replacing $\theta_r(\hat{y}_r; x, w, \lambda)$ with $\theta_r(\hat{y}_r; x, w, \lambda) + \ell_r(\hat{y}_r, y_r)$. Letting $z = (x, y)$ we introduce $\theta_r(\hat{y}_r; z, w, \lambda) = \theta_r(\hat{y}_r; x, w, \lambda) + \ell_r(\hat{y}_r, y_r)$.

The norm-product approach for inference may use counting numbers c_r to control the peakedness of the beliefs, namely

$$b_r(\hat{y}_r|x; w, \lambda, c) \propto \exp\big((\theta_r(\hat{y}_r; x, w, \lambda) + \ell_r(\hat{y}_r, y_r))/c_r\big). \tag{10}$$

As $c_r \to 0$ this distribution approaches a zero-one probably around the most likely structure, i.e., the desired loss adjusted prediction. Thus the following program blends learning with loss adjusted inference, as well as structured predictions (cf. Meshi et al. (2010)):

Theorem 5 *Consider the loss adjusted beliefs given in Equation (10) and their maximum likelihood concave program:*

$$\max_{w,\lambda} \sum_{z \in S} \sum_{r \in \mathcal{R}} c_r \cdot \log b_r(y_r|x; w, \lambda, c) - \frac{C}{2}\|w\|^2.$$

Set $\hat{\theta}_r(y_r; z, w) = \theta_r(y_r; x, w) + \ell_r(y_r, \hat{y}_r)$. *Then, blending the following loss adjusted learning and inference update rules is guaranteed to converge to the programs optimal value for any*

$c_r > 0$.

$$\mu_{p\to r}(y_r;x) = c_p \log\Big(\sum_{y_p\setminus y_r} \exp\big((\hat\theta_p(y_p;z,w) + \sum_{c\in C(p)\setminus r}\lambda_{c\to p}(y_c;x) - \sum_{p'\in P(p)}\lambda_{p\to p'}(\hat y_p;x))/c_p\big)\Big)$$

$$\lambda_{r\to p}(y_r;x) = \frac{c_r\Big(\hat\theta_r(y_r;z,w) + \sum_{c\in C(r)}\lambda_{c\to r}(y_c;x) + \sum_{p'\in P(r)}\mu_{p'\to r}(y_r;x)\Big)}{c_r + \sum_{p\in P(r)} c_p} - \mu_{p\to r}(y_r;x)$$

$$w_k \;\leftarrow\; w_k - \eta\Big(\sum_{(x,y)\in\mathcal{S}}\sum_{r\in\mathcal{R}_k}\Big(\sum_{\hat y_r\in\mathcal{Y}_r} b_r(\hat y_r|x;w,\lambda,c)\phi_{k,r}(x,\hat y_r) - \phi_{k,r}(x,y_r)\Big) + Cw_k\Big).$$

Moreover, the beliefs converge to consistent beliefs that are the unique solution of the dual program

$$\max_{b_r,u}\sum_{(x,y),r}\Big(c_r H(b_r) + \sum_{\hat y_r} b_r(\hat y_r|x)\ell_r(y_r,\hat y_r)\Big) - \frac{1}{2C}\|u\|^2,$$

subject to $u_k = \sum_{(x,y),r}\big(\sum_{\hat y_r} b_r(\hat y_r|x)\phi_{k,r}(x,\hat y_r) - \phi_{k,r}(x,y_r)\big)$, $b_r(\hat y_r|x)\geq 0$, $\sum_{\hat y_r} b_r(\hat y_r|x) = 1$ *and* $b_r(\hat y_r|x) = \sum_{\hat y_p\setminus\hat y_r} b_p(\hat y_p|x)$.

Proof The update rule for w follows from Lemma 3 and the update rule for λ follows from Lemma 2. Since the program is unconstrained, the optimal λ is attained when the gradient vanishes, or equivalently $\sum_{y_p\setminus y_r} b_p(y_p|x;w,\lambda) = b_r(y_r|x;w,\lambda)$, while $b_r(\cdot)$ are the reparametrized beliefs. When setting $\mu_{p\to r}(y_r;x)$ as above, the marginalization of $b_p(y_p|x;w,\lambda)$ satisfy $\sum_{y_p\setminus y_r} b_p(\hat y_p|x;w,\lambda)\propto\exp\big((\mu_{p\to r}(y_r;x)+\lambda_{r\to p}(y_r;x))/c_p\big)$. Therefore, by taking the logarithm, the gradient vanishes whenever the beliefs numerators agree

$$\frac{\mu_{p\to r}(y_r;x)+\lambda_{r\to p}(y_r;x)}{c_p} = \frac{\theta_r(y_r;z,w) + \sum_{c\in C(r)}\lambda_{c\to r}(y_c;x) - \sum_{p\in P(r)}\lambda_{r\to p}(y_r;x)}{c_r}$$

Multiplying both sides by $c_r c_p$ and summing both sides with respect to $p'\in P(r)$ we are able to isolate $\sum_{p'\in P(r)}\lambda_{r\to p'}(y_r;x)$. Plugin it into the above equation results in the desired inference update rule, i.e., $\lambda_{r\to p}(y_r;x)$ for which the partial derivatives vanish.

To prove the duality theorem we show that the primal program is the dual of its dual program using its Lagrangian. For every $r,\hat y_r, p\in P(r)$ we introduce the Lagrange multipliers $\lambda_{r\to p}(\hat y_r;x)$ for the marginalization constraints $b_r(\hat y_r|x) = \sum_{\hat y_p\setminus\hat y_r} b_p(\hat y_p|x)$. We also introduce the Lagrange multipliers w_k for the constraints $u_k = \sum_{(x,y),r}\big(\sum_{\hat y_r} b_r(\hat y_r|x)\phi_{k,r}(x,\hat y_r) - \phi_{k,r}(x,y_r)\big)$. We let Δ_r refer to the probability simplex constraining the beliefs $b_r(\hat y_r|x)$. Then the primal program takes the form $p(\lambda,w) = \max_{b_r\in\Delta_r,u} L(b,u,\lambda,w)$ which decomposes to $\sum_{z,r} c_r \max_{b_r\in\Delta_r}\{H(b_r) + \sum_{\hat y_r} b_r(y_r|x)(\ell_r(y_r,\hat y_r) + \theta_r(\hat y_r;x,w,\lambda))/c_r\} + w^\top d + \max_u\{w^\top u - \frac{1}{2C}\|u\|^2\}$, where d_k are the empirical moments $d_k = \sum_{(x,y),r}\phi_{k,r}(x,y_r)$. Thus the primal is the sum of conjugate dual functions. The primal is then derived since the log-partition function is the conjugate dual of the entropy function and the conjugate dual of $\frac{1}{2C}\|u\|^2$ is $\frac{C}{2}\|w\|^2$. The form in the theorem is obtained since $\ell_r(y_r,y_r) = 0$ and $\sum_r\sum_{p\in P(r)}\lambda_{r\to p}(y_r;x) - \sum_r\sum_{c\in C(r)}\lambda_{c\to r}(y_c;x)\equiv 0$. Finally, the convergence properties are a consequence of Tseng and Bertsekas (1987) similarly to Theorem 4. ■

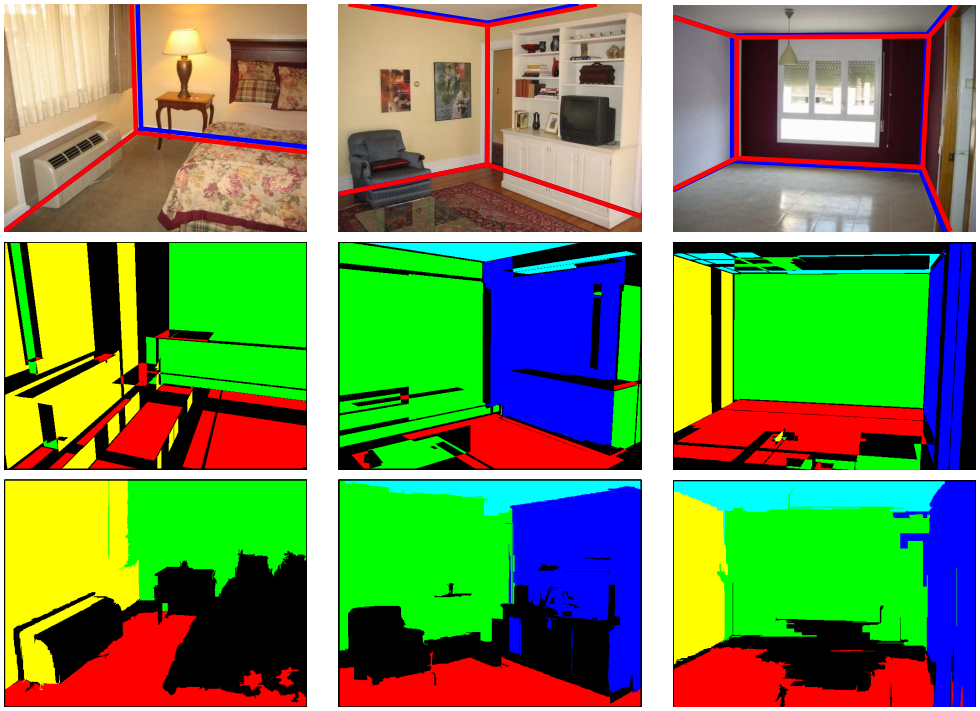

Figure 3: The top row shows three indoor scenes from the layout dataset (Hedau et al., 2009). Orientation maps (Lee et al., 2009) and geometric context (Hoiem et al., 2007) features for the respective images are illustrated in the center and bottom row respectively.

Note that the log-beliefs are balanced by c_r, to compensate for their exponent peakedness. This balancing makes the objective well defined at $c_r = 0$ as the limit of $c_r \to 0$. In this case the program is non-smooth and one needs to consider the sub gradient in the form of max-beliefs. However, this setting was already developed, although in the different context of structured-SVMs using dual losses, and we refer the interested reader to Meshi et al. (2010) for more details.

6. Experiments

The effectiveness of the discussed framework was recently illustrated by employing this algorithm as the learning engine for various computer vision tasks: scene understanding (Yao et al. (2012); Lin et al. (2013)), shape reconstruction (Salzmann and Urtasun (2012)) indoor scene understanding (Schwing et al. (2012a); Schwing and Urtasun (2012)), depth estimation (Yamaguchi et al. (2012)), flow estimation (Yamaguchi et al. (2013)) and visual-language understanding (Fidler et al. (2013)). The code is publicly available on `http://www.alexander-schwing.de/projectsGeneralStructuredPredictionLatentVariables.php`.

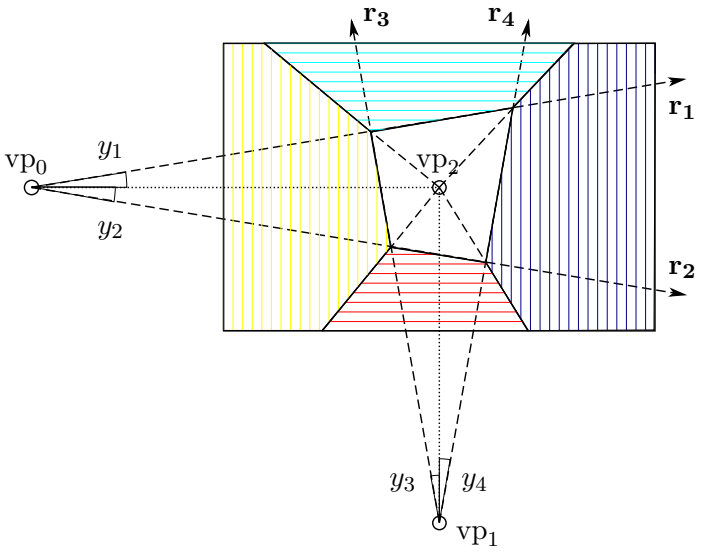

Figure 4: Ray parameterization of the layout given detected vanishing points. A state of a discretized variable y_i gives rise to a particular ray $\mathbf{r_i}$ originating from a detected vanishing point. Four rays are sufficient to define a room layout, i.e., the location of the walls.

In the following we demonstrate the various properties of blending learning and inference. For this purpose we elaborate more carefully on a 3D scene understanding application (Schwing et al., 2012a) evaluated on the well known layout dataset (Hedau et al., 2009) containing 314 indoor images. Given a single image similar to the ones illustrated in the top row of Figure 3, we aim at estimating the location of the left, front and right walls as well as ceiling and floor. Formulating this task as a pixelwise semantic segmentation application permits a large number of configurations that are physically not plausible. We therefore constrain the space of configurations by parametrizing the application using three dominant vanishing points as illustrated in Figure 4. The random variables $y_1, \ldots, y_4$ correspond to discretized angles of rays, each one originating from a detected vanishing point. Such a parametrization in terms of rays permits only physically plausible configurations and assumes the world to be Manhattan like, i.e., the observed room is aligned according to the three dominant directions.

To obtain the dominant directions we employ the vanishing point detector of Hedau et al. (2009). Due to the involved randomness it failed in our case on 9 training images and was successful on all 105 test set instances. For image x and a given layout hypothesis $y = (y_1, \ldots, y_4)$ we compute a 55-dimensional feature vector $\phi(x, y)$ which is constructed based on geometric context (Hoiem et al., 2007) and orientation maps (Lee et al., 2009) illustrated in the middle and bottom row of Figure 3 respectively.

More specifically, for each of the five hypothesized wall areas (note that the back wall is never observed) obtained from projecting the predicted layout y into the image, we count how frequently geometric context estimates the five different wall labels plus an additional clutter label within each hypothesized wall. This gives rise to a $5 \cdot 6 = 30$ dimensional

c_r \ C	1e2	1	1e-2	1e-4	1e-6
10.0000	30.92	17.53	13.81	14.24	14.46
1.0000	23.95	16.26	14.46	14.86	14.86
0.1000	17.64	13.69	14.83	14.80	14.80
0.0100	15.83	**13.59**	14.20	14.25	14.25
0.0010	15.46	13.82	14.00	13.85	13.85
0.0001	16.04	14.09	13.95	13.96	13.97
0	15.72	13.70	13.82	13.91	13.98

c_r \ C	1e2	1	1e-2	1e-4	1e-6
10.0000	27.28	19.54	17.43	**16.46**	16.49
1.0000	22.66	17.87	16.55	16.83	16.83
0.1000	19.41	17.43	16.74	16.91	16.96
0.0100	17.98	16.48	17.04	16.86	16.86
0.0010	17.92	17.18	16.95	16.83	16.87
0.0001	17.95	17.06	17.07	16.80	16.71
0	18.01	16.97	17.25	17.04	17.01

Figure 5: Left: Test set percentage pixel error on the layout data set Hedau et al. (2009). Right: Test set pixel classification error in % on the bedroom data set of Hedau et al. (2010).

feature vector. Similarly for orientation map evidence we count how many of the five possible labels fell onto each of the five hypothesized wall areas, resulting in a $5 \cdot 5 = 25$ dimensional vector. Combining both image evidences we hence obtain a feature vector ϕ having a total of $K = 55$ entries, each being represented by a graphical model having nodes $\mathcal{R}_k$, $k \in \{1, \dots, K\}$.

Note that a vanilla implementation of these color counting features results in potentials of order four for the front wall, and of order three for all other walls. High-order potentials increase the complexity of learning and inference, therefore, the exact decomposition of those high-order potentials into pairwise terms using 'integral geometry' Schwing et al. (2012a) is important for tractability.

In our experiments we learn and infer with the same counting numbers. For example, when we learn log-beliefs by setting the counting numbers to 1, we also infer with log-beliefs at test time, namely setting the counting numbers to 1. Figure 5 demonstrates the tradeoffs of learning with various counting numbers. This experiment also compares to blending of structured-SVMs of Meshi et al. (2010) when setting the counting numbers to 0.

In Figure 6 we illustrate the optimized cost function, i.e., the surrogate training loss over wall clock time. We observe that our discussed blended learning approach ('Ours') converges quickly, i.e., in less than 50 seconds, to a zero primal dual gap. In contrast it takes the standard learning approach ('Standard 20'), which performs 20 message passing iterations, more than 600 seconds to converge to the same solution. Hence blended learning is more than 10 times faster on this example. Next we investigate whether a standard message passing approach with 20 iterations is overly pessimistic for this dataset. To this end we use 10 message passing iterations and refer to the obtained result as 'Standard 10.' Investigating Figure 6 more carefully we observe that 10 message passing iterations are not sufficient to close the duality gap, i.e., the approach does never converge to the same solution.

These plots validate our theoretical results. However, often we are interested in the resulting test set error. Therefore we compare learning which uses the proposed blending approach with the standard technique using the test set performance in Figure 7. More specifically, in Figure 7(a) we illustrate the test set performance of blending with all counting numbers equal to one ('Blending (1)') and compare it to the test set performance of blending with all counting numbers equal to zero ('Blending (0)') derived by Meshi et al.

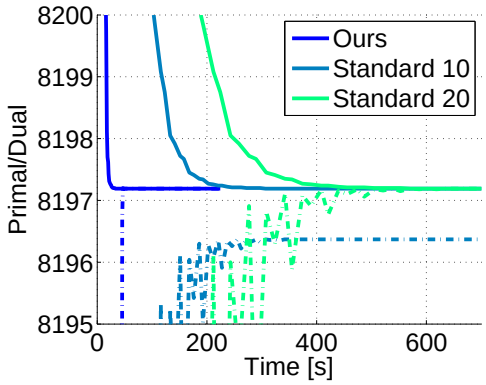

Figure 6: Primal surrogate loss (dashed) and its dual (solid) over wall clock time measured in seconds for blended learning, the standard approach with 10 iterations, and the standard approach with 20 iterations.

(2010), the test set performance of convex learning with 20 message passing iterations with both counting numbers equal to zero (cf. Meshi et al. (2010)) and one, i.e., 'Standard 20 (0)' and 'Standard 20 (1)' respectively. We observe that the test set accuracy drops significantly faster which is expected since we are able to update the parameter vector more frequently. Similar to the aforementioned surrogate training loss result we observe a performance improvement of more than one order of magnitude, i.e., we obtain accurate test set results more than 10 times faster.

In a next experiment we evaluate the importance of the loss function. The results are illustrated in Figure 7(b). We observe the loss to be important for the case where the counting numbers equal zero. We are able to obtain a performance similar to loss-included setting if the counting numbers are equal to one. However algorithms which do not use a loss function while having counting numbers equal to zero got stuck prematurely and are hence not even visible in Figure 7(b), where the scale was adjusted to fit the other plots.

Next we observe the behavior if we reduce the number of iterations for standard learning from 20 down to 2. Recall, from the surrogate training loss results that the primal-dual gap will not reach zero in this setting. The test set errors are illustrated in Figure 7(c). We observe that the standard method is approaching the blending technique. This indicates that a primal-dual gap is not necessarily important for good generalization performance.

In Figure 7(d) we illustrate that blending offers a wide range of possibilities. Indeed, we are not restricted to performing only a single message passing iteration before updating the parameter vector. Rather any arbitrary scheduling is possible. As illustrated in the results we observe slightly faster convergence when updating the messages twice as frequently as the parameter vector, i.e., we perform two rounds of message passing updates before updating the parameters of the model. This is expected since the distribution is more accurate while message passing updates are quick in this setup. We refer the interested reader to (Schwing et al., 2012a) for additional results exploring counting numbers other than zero and one.

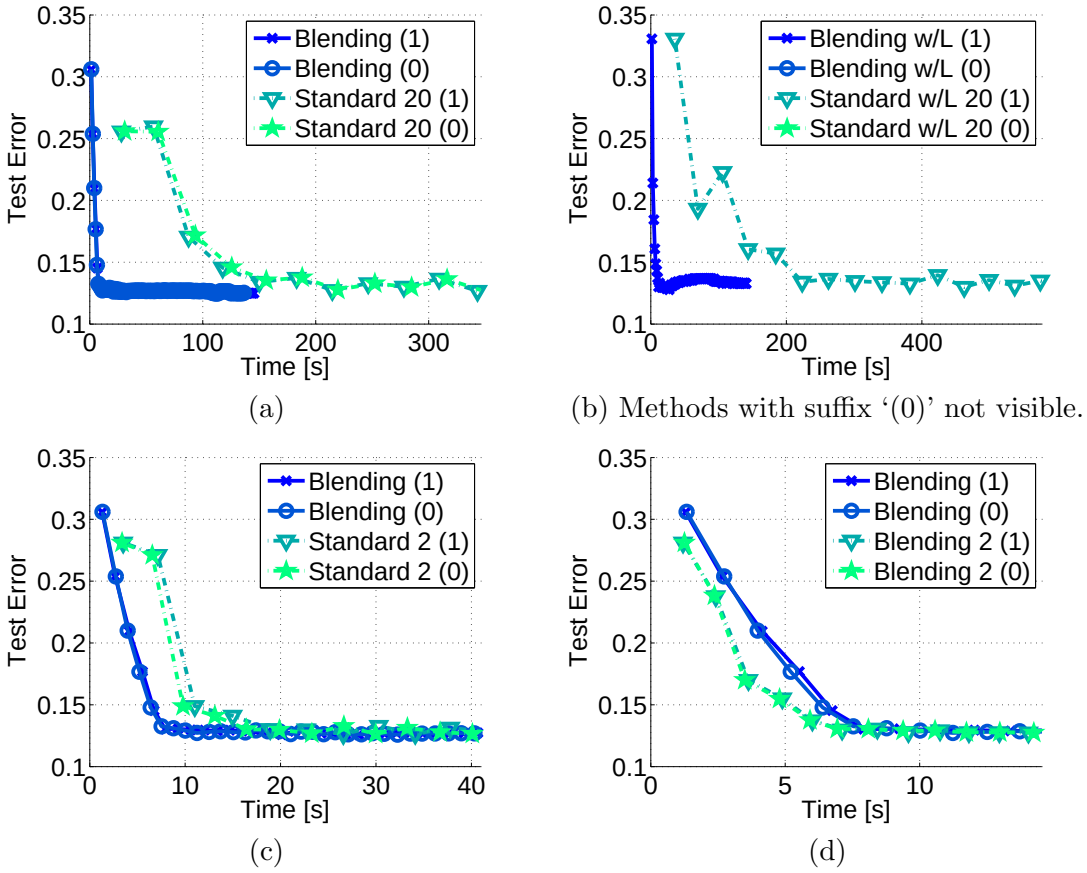

Figure 7: A comparison of the test set error over training time for different parameter settings. In (a) we compare standard learning with blended learning when including a loss function. In (b) we provide the comparison when not using a loss function. Note that using MAP estimation without loss got stuck at a high error. The results in (c) demonstrate the effects when reducing the iterations of the standard learning method, while (d) illustrates the flexibility offered by blending where we use two message passing iterations before updating the parameters.

7. Conclusion and Discussion

In this paper we describe the objective function for nested learning and inference in CRFs, for which approximate inference algorithms are used as a sub-procedure. The devised objective maximizes the log-beliefs — probability distributions over subsets of variables that agree on their marginal probabilities. This objective is concave and consists of two types of variables that are related to the learning and inference tasks respectively. Therefore, we are able to blend the learning and inference procedures and effectively get to the optimum much faster than nested learning and inference approach, which uses inference algorithms as black-box solvers. We show the computational advantage for using blended algorithms over nested ones. We also provide an efficient C++ implementation with a Matlab wrapper.

This work extends Hazan and Urtasun (2010) while simplifying its theoretical and practical concepts. Specifically, we introduce the learning program as maximizing log-beliefs, explain the relations between nested and blended learning-inference and derives a blending algorithm for general graphs. This work is extended to latent variables (Schwing et al. (2012b)) and deep learning (Chen* et al. (2015); Schwing and Urtasun (2015)). The effectiveness of the discussed framework was recently illustrated by employing this algorithm as the learning engine for various computer vision tasks: 2D scene understanding (Yao et al. (2012)), 3D scene understanding (Lin et al. (2013)), shape reconstruction (Salzmann and Urtasun (2012)) indoor scene understanding (Schwing et al. (2012a); Schwing and Urtasun (2012)), depth estimation (Yamaguchi et al. (2012)), flow estimation (Yamaguchi et al. (2013)) and visual-language understanding (Fidler et al. (2013)). We believe it is interesting to show in the future if this algorithm provides state-of-the-art performance in domains other than computer vision, or whether the statistics in computer vision are used by this approach in a special manner.

The computational complexity of our algorithm depends on norm-products over the labels of regions. Therefore, efficient techniques over large regions in inference can be applied as sub-procedure in our algorithm, e.g., Kohli et al. (2009); Batra et al. (2010); Tarlow et al. (2010, 2011); Tarlow and Zemel (2012).

In our framework, we can enforce the moment matching constraints through general concave functions. These function translate to a regularization in the primal. For computational efficiency we choose the square function but we did not investigate the different moment matching and regularization functions. Moreover, we enforce the marginalization constraints through indicator functions, in order to obtain closed-form solution in the primal block coordinate descent. However, we have shown that using the penalty method we can enforce the marginalization constraints with different convex functions. Further explorations of structured prediction with squared penalties appears in work by Meshi et al. (2015). We leave the affect of general convex functions on moment matching and regularization, as well as marginalization constraints and efficient message-passing to future research.

Interestingly, our approach confirms that the parameters of graph based structured predictors can be efficiently learned in many real-life problems. This validates the intuition behind the theoretical results of Wainwright et al. (2003); Wainwright (2006) which asserts that whenever learning and inference occur together one can use pseudo moment matching for learning the parameters. This concept was put forward in the general framework of learning to reason by Khardon and Roth (1997) and we leave for future research to find different frameworks which have similar learning-prediction robustness.

8. Extensions: regularizations and the penalty method

Duality theory turned out to be very effective in machine learning as it provides a principled way to decompose the different ingredients of the primal objective through its Lagrange multipliers. The dual decomposition in turn provides the means to efficiently estimate the different ingredients while maintaining their consistency using the dual objective.

When dealing with convex programs one usually needs to consider the set of primal feasible solutions while constructing the dual function. We find it simpler to describe

the primal program using extended real-valued convex functions, which are functions that can take on the value of infinity. By using extended real-valued functions we can ignore their domains, i.e., points for which a function attains the value other than infinity, thus simplifying the derivations. The dual programs of extended real valued convex functions $g(\mu)$ are conveniently formulated in terms of their conjugate dual

$$g^*(z) = \max_{\mu} \Big\{ \mu^\top z - g(\mu) \Big\}.$$

Throughout this work we use the following duality theorem, known as the Fenchel duality (cf. Fenchel (1951); Rockafellar (1970); Bertsekas et al. (2003)):

Theorem 6 *Let $f(\cdot), h_1(\cdot), h_2(\cdot)$ be extended real-valued, continuous and convex functions. The following are primal and dual programs:*

$$\begin{aligned}
\min_{\lambda,w} \quad & \sum_{(x,y)\in S, r\in\mathcal{R}} f\Big(\theta_r(\cdot; x, w, \lambda) + \ell_r(\cdot, y_r)\Big) + \sum_k h_1(w_k) + \sum_{(x,y),r,\hat{y}_r,p\in P(r)} h_2(\lambda_{r\to p}(\hat{y}_r; x)) \\
\max_{b_r} \quad & \sum_{(x,y)\in S, r\in\mathcal{R}} \Big(- f^*(b_r) + \sum_{\hat{y}_r} b_r(\hat{y}_r|x)\ell_r(y_r, \hat{y}_r)\Big) \\
& - \sum_k h_1^*\Big(\sum_{(x,y),r} \big(\sum_{\hat{y}_r} b_r(\hat{y}_r)\phi_{k,r}(x, \hat{y}_r) - \phi_{k,r}(x, y_r)\big)\Big) \\
& - \sum_{(x,y),r,\hat{y}_r,p} h_2^*\Big(\sum_{\hat{y}_p\setminus\hat{y}_r} b_p(\hat{y}_p|x) - b_r(\hat{y}_r|x)\Big)
\end{aligned}$$

Proof The proof goes along the lines of Theorem 5. The dual program takes the form

$$\begin{aligned}
& \sum_{(x,y)\in S, r\in\mathcal{R}} \Big(- f^*(b_r) + \sum_{\hat{y}_r} b_r(\hat{y}_r|x)\ell_r(y_r, \hat{y}_r)\Big) - \sum_k h_1^*(u_k) - \sum_{(x,y),r,\hat{y}_r,p} h_2^*(\delta_{r\to p}(\hat{y}_r; x)) \\
& s.t. \quad u_k = \sum_{(x,y),r} \big(\sum_{\hat{y}_r} b_r(\hat{y}_r|z)\phi_{k,r}(x, \hat{y}_r) - \phi_{k,r}(x, y_r)\big) \\
& \qquad \delta_{r\to p}(\hat{y}_r; x) = \sum_{\hat{y}_p\setminus\hat{y}_r} b_p(\hat{y}_p|x) - b_r(\hat{y}_r|x)
\end{aligned}$$

Constructing its Lagrangian with the Lagrange multipliers $w_k, \lambda_{r\to p}(\hat{y}_r; x)$, we obtain the primal program $p(w, \lambda) = \max_{b,u,d} L(b_r, \lambda, w)$ which decomposes to $\sum_{(x,y),r} \max_{b_r}\{-f^*(b_r) + \sum_{\hat{y}_r} b_r(y_r|x)(\ell_r(y_r, \hat{y}_r) + \theta_r(\hat{y}_r; x, w, \lambda))\} + \sum_{(x,y),r,\hat{y}_r,p} \max_\delta\{\delta_{r\to p}(\hat{y}_r)\lambda_{r\to p}(\hat{y}_r) - h_2^*(\delta_{r\to p}(\hat{y}_r))\} + w^\top e + \sum_k \max_{u_k}\{w_k u_k - h_1^*(u_k)\}$, where $e_k = \sum_{(x,y),r} \phi_{k,r}(x, y_r)$. The result then follows since the conjugate dual of $f^*(), h_1^*(), h_2^*()$ are $f(), h_1(), h_2()$ respectively. ■

The above formulation describes a dual program with a selection rule $f^*(\cdot)$, a penalty function for learning moment matchings $h_1^*(\cdot)$ and a penalty function $h_2^*(\cdot)$ for fitting the marginalization constraints. Although these penalty functions are conceptually different — $h_1^*(\cdot)$ relates to learning the parameters w and $h_2^*(\cdot)$ relates to inferring the marginal probabilities — they have the same variational interpretation.

The primal program translates the dual penalty functions $h_1^*(\cdot), h_2^*(\cdot)$ to regularization functions $h_1(\cdot)$, $h_2(\cdot)$. Whenever the primal functions are smooth we can use the chain rule to derive the primal program gradients:

$$\begin{aligned}
\frac{\partial}{\partial w_k} \quad &: \quad \sum_{(x,y)\in\mathcal{S}} \sum_{r\in\mathcal{R}} \Big(\sum_{\hat{y}_r} \frac{\partial f(\theta_r(\cdot;z,w,\lambda))}{\partial \theta_r(\hat{y}_r;z,w,\lambda)} \phi_{k,r}(x,\hat{y}_r) - \phi_{k,r}(x,y_r) \Big) + \nabla h_k(w_k) \\
\frac{\partial}{\partial \lambda_{r\to p}(\hat{y}_r;x)} \quad &: \quad \sum_{\hat{y}_p \setminus \hat{y}_r} \frac{\partial f(\theta_p(\cdot;z,w,\lambda))}{\partial \theta_p(\hat{y}_p;z,w,\lambda)} - \frac{\partial f(\theta_r(\cdot;z,w,\lambda))}{\partial \theta_r(\hat{y}_r;z,w,\lambda)} + \nabla h_2(\lambda_{r\to p}(\hat{y}_r;x))
\end{aligned}$$

The above generalizes the learning-inference blending algorithm for general functions $f(\cdot)$ and regularizations $h_1(\cdot), h_2(\cdot)$. The power of setting $f(\cdot)$ to be the log-partition function is that its derivatives are beliefs therefore we obtain an intuitive probabilistic interpretations. The parameters derivatives result in moment matching constraints

$$\sum_{(x,y)\in\mathcal{S}} \sum_{r\in\mathcal{R}} \Big(\sum_{\hat{y}_r} \frac{\partial f(\theta_r(\cdot;z,w,\lambda))}{\partial \theta_r(\hat{y}_r;z,w,\lambda)} \phi_{k,r}(x,\hat{y}_r) - \phi_{k,r}(x,y_r) \Big).$$

The re-parametrization derivatives with respect to the messages $\lambda_{r\to p}(y_r;x)$ are then marginalization constraints

$$\sum_{\hat{y}_p \setminus \hat{y}_r} \frac{\partial f(\theta_p(\cdot;z,w,\lambda))}{\partial \theta_p(\hat{y}_p;z,w,\lambda)} - \frac{\partial f(\theta_r(\cdot;z,w,\lambda))}{\partial \theta_r(\hat{y}_r;z,w,\lambda)}$$

Moreover, whenever $h_2(\cdot) \equiv 0$, we are able to derive closed-form update rules.

Acknowledgments

We are most grateful to David McAllester for helpful discussions, and the anonymous reviewers for their helpful comments. This work was supported in part by the German-Israeli Foundation grant I-1123-407.6-2013.

References

S. M. Aji and R. J. McEliece. The generalized distributive law and free energy minimization. In *Allerton*, 2001.

D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A.Y. Ng. Discriminative learning of markov random fields for segmentation of 3D range data. In *Proc. CVPR*, 2005.

D. Batra, A. C. Gallagher, D. Parikh, and T. Chen. Beyond trees: MRF inference via outer-planar decomposition. In *Proc. CVPR*, 2010.

D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

L.-C. Chen*, A. G. Schwing*, A. L. Yuille, and R. Urtasun. Learning Deep Structured Models. In *Proc. ICML*, 2015. * equal contribution.

M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. ACL*, 2002.

J. Domke. Parameter learning with truncated message-passing. In *Proc. CVPR*, 2011.

P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2010.

W. Fenchel. *Convex cones, sets, and functions*. Princeton University, Department of Mathematics, 1951.

S. Fidler, A. Sharma, and R. Urtasun. A sentence is worth a thousand pixels. In *Proc. CVPR*, 2013.

T. Hazan and A. Shashua. Convergent message-passing algorithms for inference over general graphs with convex free energies. In *Proc. UAI*, 2008.

T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *Information Theory*, 2010.

T. Hazan and R. Urtasun. A Primal-Dual Message-Passing Algorithm for Approximated Large Scale Structured Prediction. In *Proc. NIPS*, 2010.

V. Hedau, D. Hoiem, and D. Forsyth. Recovering the Spatial Layout of Cluttered Rooms. In *Proc. ICCV*, 2009.

Varsha Hedau, Derek Hoiem, and David Forsyth. Thinking Inside the Box: Using Appearance Models and Context Based on Room Geometry. In *Proc. ECCV*, 2010.

T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research*, 2006.

D. Hoiem, A. A. Efros, and M. Hebert. Recovering Surface Layout from an Image. *IJCV*, 2007.

R. Khardon and D. Roth. Learning to reason. *Journal of the ACM (JACM)*, 1997.

P. Kohli, L. Ladickỳ, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. *IJCV*, 2009.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Proc. EMNLP*, 2010.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001.

S. L. Lauritzen. *Graphical models*, volume 17. Oxford University Press, USA, 1996.

G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. *Proc. NIPS*, 2002.

D. C. Lee, M. Hebert, and T. Kanade. Geometric Reasoning for Single Image Structure Recovery. In *Proc. CVPR*, 2009.

A. Levin and Y. Weiss. Learning to Combine Bottom-Up and Top-Down Segmentation. In *Proc. ECCV*, 2006.

D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *Proc. ICCV*, 2013.

T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms-a unifying view. In *Proc. UAI*, 2009.

O. Meshi, A. Jaimovich, A. Globerson, and N. Friedman. Convexifying the bethe free energy. In *Proc. UAI*, 2009.

O. Meshi, D. A. Sontag, T. S. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *Proc. ICML*, 2010.

O. Meshi, N. Srebro, and T. Hazan. Efficient Training of Structured SVMs via Soft Constraints. In *Proc. AISTATS*, 2015.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer, 2004.

R.T. Rockafellar. *Convex analysis*. Princeton university press, 1970.

M. Salzmann and R. Urtasun. Beyond Feature Points: Structured Prediction for Monocular Non-rigid 3D Reconstruction. In *Proc. ECCV*, 2012.

A. G. Schwing and R. Urtasun. Efficient exact inference for 3d indoor scene understanding. In *Proc. ECCV*, 2012.

A. G. Schwing and R. Urtasun. Fully Connected Deep Structured Networks. http://arxiv.org/abs/1503.02351, 2015.

A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Distributed message passing for large scale graphical models. In *Proc. CVPR*, 2011.

A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction for 3d indoor scene understanding. In *Proc. CVPR*, 2012a.

A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient Structured Prediction with Latent Variables for General Graphical Models. In *Proc. ICML*, 2012b.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Proc. UAI*, 2008.

C. Sutton and A. McCallum. Piecewise training for structured prediction. *Machine Learning*, 2009.

R. Tappenden, P. Richtárik, and J. Gondzio. Inexact coordinate descent: complexity and preconditioning. *arXiv preprint arXiv:1304.5530*, 2013.

D. Tarlow and R. S. Zemel. Structured output learning with high order loss functions. In *Proc. AISTATS*, 2012.

D. Tarlow, I. E. Givoni, and R. S. Zemel. Hopmap: Efficient message passing with high order potentials. In *Proc. AISTATS*, 2010.

D. Tarlow, D. Batra, P. Kohli, and V. Kolmogorov. Dynamic tree block coordinate ascent. *Proc. ICML*, 2011.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. *Proc. NIPS*, 2004.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proc. ICML*, 2005.

P. Tseng and D.P. Bertsekas. Relaxation methods for problems with strictly convex separable costs and linear constraints. *Mathematical Programming*, 38(3):303–321, 1987.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML*, 2004.

M. J. Wainwright. Estimating the Wrong Graphical Model: Benefits in the Computation-Limited Setting. *JMLR*, 2006.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 2008.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *Proc. Workshop on AI and Stats.*, 2003.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *Trans. on Information Theory*, 51(7):2313–2335, 2005.

K. Yamaguchi, T. Hazan, D. McAllester, and R. Urtasun. Continuous markov random fields for robust stereo estimation. In *Proc. ECCV*, 2012.

K. Yamaguchi, D. McAllester, and R. Urtasun. Robust monocular epipolar flow estimation. In *Proc. CVPR*, 2013.

C. Yanover, O. Schueler-Furman, and Y. Weiss. Minimizing and learning energy functions for side-chain prediction. In *RECOMB*. Springer, 2007.

J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Proc. CVPR*, 2012.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory*, 2005.